UNIVERSIDAD DE QUINTANA ROO
DIVISIÓN DE CIENCIAS E INGENIERÍA

Caracterización de la Pérdida de Paquetes en Tráfico de Video Streaming

TRABAJO DE TESIS
PARA OBTENER EL GRADO DE
Ingeniero en Redes

PRESENTA
Br. Christian Jesús Vadillo Mejía

DIRECTOR DE TESIS
Dr. Homero Toral Cruz

ASESORES

Ing. Francisco Méndez Martínez

Dr. Freddy Ignacio Chan Puc

Dr. Jaime Dionisio Cuevas Domínguez

Dr. Víctor Soberanis Cruz

CHETUMAL QUINTANA ROO, MÉXICO, DICIEMBRE DE 2016

UNIVERSIDAD DE QUINTANA ROO
DIVISIÓN DE CIENCIAS E INGENIERÍA

TRABAJO DE TESIS ELABORADO BAJO SUPERVISIÓN DEL COMITÉ DE ASESORÍA Y APROBADO COMO REQUISITO PARCIAL PARA OBTENER EL GRADO DE:

INGENIERO EN REDES

Universidad de Quintana Roo
DCI DIVISIÓN DE CIENCIAS E INGENIERÍA

Comité de Trabajo de Tesis

DIRECTOR:

Dr. Homero Toral Cruz

ASESOR:

Ing. Francisco Méndez Martínez

ASESOR:

Dr. Freddy Ignacio Chan Puc

UNIVERSIDAD DE QUINTANA ROO
SERVICIOS ESCOLARES
TITULACIONES

CHETUMAL QUINTANA ROO, MÉXICO, DICIEMBRE DE 2016

Dedicatoria

Dedico este trabajo y todo el proceso llevado a cabo para la finalización de esta etapa académica a mi mayor tesoro, mi familia. Juntos hemos vividos buenos y malos momentos, pero siempre acompañándonos en todas las etapas de la vida.

Una dedicatoria especial a mi madre, por su incondicional apoyo e inspiración a dar lo mejor de mí.

Agradecimientos

Agradezco primero a Dios y a la vida por darme la oportunidad de vivir esta experiencia inigualable.

A mi familia que me ha dado todo en mi vida, por la paciencia que tuvieron conmigo y amor infinito, apoyándome siempre en los momentos difíciles, a pesar de saber lo complicado de mi actitud, y mi falta de empatía. Quiero demostrarles que sé apreciar sus gestos, a pesar de mi silencio.

Quiero agradecer solo a 3 profesores que particularmente se ganaron mi respeto y admiración. A mi director de tesis, el Dr. Homero Toral Cruz, por su tiempo, sus sabios consejos y sus constantes retos para ayudarme a explotar más mis aptitudes. No creo que haya podido haber una mejor persona para guiarme durante todo este proceso del desarrollo de mi tesis.

Al Dr. Inocente Bojórquez Báez, porque fue la primera persona que pudo ver a través de mí, me enseño cuales eran mis debilidades y mis fortalezas.

A la M.T.I. Melissa Blanqueto Estrada, por su paciencia y dedicación, por ser un foco de inspiración. Cuando entré a esta carrera mis conocimientos se limitaban al uso de Word, Excel y PowerPoint, sin embargo, con su pericia y retos, puedo hoy decir que aprendí a las herramientas básicas para desenvolverme con éxito en el ámbito de la programación.

Resumen

Actualmente vivimos en un mundo que gira en torno a la tecnología, donde más de 3 mil millones de personas ya cuentan con acceso a Internet, con el auge de los dispositivos móviles y aunado a la constante mejora de la infraestructura y ancho de banda para los usuarios finales, esta cifra está por crecer inimaginablemente.

Producto de estos cambios, se observan nuevas tendencias en el tráfico multimedia de Internet, tal como las aplicaciones de video. En los últimos años el servicio de video streaming se ha popularizado gracias a plataformas como Youtube, Vimeo y Netflix. De igual forma, se ha convertido en una herramienta importante para grandes empresas e instituciones, tanto gubernamentales, como académicas, permitiéndoles expandir sus servicios.

Cuando hablamos de video streaming, es necesario tener en cuenta el papel que juega la calidad de servicio (QoS); la cual puede ser definida como la capacidad de una red para proporcionar un servicio aceptable al usuario final. Para evaluar la QoS es necesario tomar en cuenta ciertos parámetros que influyen de manera significativa en cualquier transmisión de video por una red IP. Entre los principales parámetros encontramos la pérdida de paquetes y el jitter.

La pérdida de paquetes se presenta frecuentemente en las redes IP debido a congestiones, extinción del TTL (Time To Live), retardos extremadamente grandes, etc. Existen diversos estudios que han demostrado que la pérdida de paquetes en las redes IP se presenta a ráfagas, es decir por grupos de n-paquetes consecutivos, lo cual puede tener impactos negativos en otros parámetros de QoS como jitter y de manera general en la calidad percibida por el usuario final en el tráfico de video streaming.

Por otro lado, el jitter es la variación en el retardo entre paquetes consecutivos que pertenecen al mismo flujo en una transmisión. Debido los retardos variables de encolamiento, variación en el tamaño de paquetes, cambios de rutas, pérdida de paquetes, etc., es inevitablemente la presencia del jitter en una transmisión. Esto

suele deteriorar de manera considerable la calidad del video transmitido en una red IP.

Monitorear y caracterizar estos parámetros de QoS, es de gran importancia para conocer el estado de cierta red IP y poder atenuar sus efectos negativos mediante el uso adecuado de mecanismos como FEC (Forward Error Correction) e Interleaving en el lado del receptor.

Motivados por los puntos mencionados anteriormente, este proyecto de tesis fue centrado en la caracterización de la pérdida de paquetes en tráfico de video streaming.

TABLA DE CONTENIDO

Índice de figuras

Índice de tablas

Índice de scripts

Abreviaturas

AP	Acess Point
ARPANET	The Advanced Research Projects Agency Network
ATM	Asynchronous Transfer Mode
AVC	Advanced Video Coding
BSS	Basic Service Set
CDC	Content Delivery Network
CD-ROM	Compact Disc Read-Only Memory
CIF	Common Intermediate Forma
CSMA/CA	Carrier Sense Multiple Access with Collision Avoidance
CSRC	Contributing Source
DCT	Discrete Cosine Transform
DNS	Domain Name System
ESS	Extended Service Set
FDDI	Fiber Distributed Data Interface
FTP	File Transfer Protocol
Gbps	Gigabits per second
HDS	HTTP Dynamic Streaming
HDTV	High-Definition Television
HLS	HTTP Live Streaming
HR/DSSS	High Rate/Direct Sequence Spread Spectrum
HTTP	Hypertext Transfer Protocol
IBSS	Independent Basic Service Set
ICMP	Internet Control Message Protocol
IEEE	Institute of Electrical and Electronics Engineers
IP	Internet Protocol
ISDN	Integrated Services for Digital Network
ISO	International Organization for Standardization
ISP	Internet Service Provider
ITU-T	ITU Telecommunication Standardization Sector
LAN	Local Area Network
LLC	Logical Link Control
MAC	Medium Access Control
MAN	Metropolitan Area Network
MIMO	Multiple-input Multiple-output
MMS	Microsoft Media Server
MPEG	Moving Picture Experts Group
MPEG-DASH	Dynamic Adaptive Streaming over HTTP
MRC	Maximal Ratio Combining
NTSC	National Television System Committee
OFDM	Orthogonal Frequency Division Multiplexing
OSI	Open System Interconnection
PAL	Phase Alternating Line
PAN	Personal Area Network
PDA	Personal Digital Assistant
POP	Post Office Protocol
QCIF	Quarter Common Intermediate Format
QoE	Quality of Experience
QoS	Quality of Service
RFC	Request for Comments

RTCP	Real-Time Control Protocol
RTP	Real-Time Transport Protocol
RTSP	Real-Time Streaming Protocol
SDM	Space Division Multiplexing
SINR	Signal-To-Interference-Plus-Noise Ratio
SIP	Session Initiation Protocol
SMTP	Simple Mail Transfer Protocol
SNR	Signal-To-Noise Ratio
STBC	Space-Time Block Coding
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
URL	Uniform Resource Locator
UWB	Ultra-Wide-Band
VCR	Video Cassette Recorder
WAN	Wide Area Network
WiMAX	Worldwide Interoperability for Microwave Access
WLAN	Wireless Local Area Network

CAPÍTULO I. INTRODUCCIÓN

1.1 Antecedentes

Según el reporte anual del 2015 proporcionado por la Internet Society, una organización no gubernamental global que se dedica a asegurar que Internet permanezca abierta y transparente, se experimenta un crecimiento exponencial de usuarios de Internet, existen ya 3 mil millones de usuarios en línea (Internet Society, 2015). Debido a este crecimiento de usuarios y la constante mejora de la infraestructura y los anchos de banda, el tráfico de Internet muestra una notoria tendencia al aumento del uso de aplicaciones relativas a datos pesados o que consumen altos anchos de banda (tales como audio y video streaming). Por otro lado, el Internet proporciona un servicio de mejor esfuerzo (best-effort), es decir, no garantiza la calidad de servicio. Por tal motivo, el monitoreo de los parámetros de QoS, tales como: perdida de paquetes y jitter, se vuelve un factor fundamental para poder brindar un mejor grado de satisfacción al usuario final.

Uno de los primeros trabajos de investigación sobre la medición de los efectos de pérdidas de paquetes fue publicado en junio de 1995, por Ian E. G. Richardson bajo el nombre de *"Usage parameter control cell loss effects on MPEG video"*. La investigación está basada en el estudio del efecto de la función de control de parámetro de uso en una red ATM en tráfico de vídeo codificado bajo MPEG. Una función genérica simulada de un algoritmo de control de velocidad (GRCA) monitoreó los flujos de transmisión de las celdas y las que superaban los parámetros de conexión establecidos, se descartaban para posteriormente evaluar el efecto de estas celdas perdidas en los datos de vídeo decodificados. El análisis de los resultados permitió determinar que las pérdidas de celdas degradaban gravemente la calidad de los datos de vídeo decodificados (Richardson I. E., 1995).

En 1997, Wenwu Zhu, publicó en *IEEE First Workshop on Multimedia Signal Processing,* una investigación titulada *"Modeling and simulation of MPEG-2 video transport over ATM networks considering the jitter effect"* cuyo foco de investigación se centró en el modelado y simulación de sistemas MPEG-2 cuando un flujo de

transporte MPEG-2 se envía a través de una red ATM con presencia de jitter. El modelado y los resultados de la simulación mostraron que el jitter afecta al tamaño del buffer del decodificador y la tasa de pérdida de paquetes de manera significativa (Zhu, Hou, Wang, & Zhang, 1997).

Para el año 1999, en *IEEE 3rd Workshop on Multimedia Signal Processing*, T. Hayashi publicó su investigación titulada *"Effects of IP packet loss and picture frame reduction on MPEG1 subjective quality"*, la cual evalúa los efectos de la pérdida de paquetes IP en la reducción de calidad subjetiva en un flujo de video de MPEG-1 a fin de aclarar cómo aplicar técnicas para garantizar ciertos niveles de calidad de servicio mediante el uso del protoclo RSVP (protocolo de reserva de recursos). Las pruebas mostraron que la pérdida de paquetes fue el factor dominante en la degradación de la calidad audiovisual (Hayashi, Yamasaki, Morita, & Aida, 1999).

Otro trabajo consultado fue la investigación hecha por Zhihai He, llamado *"End-to-end video quality analysis and modeling for video streaming over IP network"*, en donde deduce un modelo analítico extremo a extremo de predicción y control de la transmisión de vídeo sobre una red IP. Este modelo mapea los parámetros QoS definidos en el nivel de conexión con calidad de presentación de vídeo real en el extremo receptor. En concreto, proporciona un análisis sobre los efectos de la pérdida de paquetes en la calidad de vídeo y relacionar la tasa de pérdida de paquetes, uno de los parámetros de calidad de servicio más importantes en el nivel de conexión, a la calidad de vídeo real (He & Chen, 2002).

1.2 Planteamiento del problema

La importancia del Internet ha crecido considerablemente en los últimos años, a tal punto que se ha convertido en una de las redes de telecomunicaciones más importantes e indispensables en la vida diaria de las personas.

Debido a la rápida evolución del Internet y al incremento en la demanda de información multimedia, una de las aplicaciones en tiempo real que ha incrementado su popularidad sobre la red IP es la transmisión de video streaming (Aurelius, Lagerstedt, & Kihl, 2011).

Aunque la transmisión de video sobre Internet es un servicio muy atractivo y muy demandado últimamente en la vida diaria, representa grandes desafíos, debido a que Internet no proporciona una garantía en cuanto a la calidad del servicio que el usuario final percibe (Wu, Hou, & Zhang, 2000) (Wu D. , Hou, Zhu, Shanz, & Peha, 2001). La calidad en la transmisión del video streaming depende de diversos parámetros, sin embargo, la pérdida de paquetes tienen un significativo impacto en la calidad de servicio (Mwela & Adebomi, 2010). Para disminuir los impactos negativos de la pérdida de paquetes sobre la calidad de servicio en la transmisión de video streaming, el uso adecuado de mecanismos como FEC (Muraoka, Masuyama, Kasahara, & Takahashi, 2009) e Interleaving (Liu, y otros, 2015) en el lado del receptor pueden ser de gran ayuda.

En el presente trabajo de tesis se realizará la caracterización temporal del parámetro pérdida de paquetes en el servicio de video streaming sobre un escenario de red controlado.

Para realizar dicha caracterización temporal, se utilizará la herramienta R-statistics, la cual permitirá estudiar los patrones de pérdida a partir de los números de secuencia, y extraer estadísticas como: número de paquetes perdidos, número de paquetes recibidos, paquetes recibidos fuera de orden o secuencia, número de paquetes perdidos de forma consecutiva, longitud de pérdidas consecutivas medida en número de paquetes, número de gaps entre paquetes perdidos de forma consecutiva, longitud de gaps medida en número de paquetes, etc.

La caracterización obtenida a partir de las estadísticas mencionadas anteriormente, puede ser usada en trabajos futuros para aplicar de manera correcta algún mecanismo como FEC o interleaving que contribuya a incrementar el desempeño en el servicio de video streaming.

1.3 Objetivo general

- Caracterizar el comportamiento temporal de la pérdida de paquetes en el servicio de video streaming y analizar su impacto en parámetros de QoS, tales como el jitter.

1.4 Objetivos particulares

- Generar tráfico de video streaming bajo los esquemas de codificación MPEG-1 y MPEG-2 mediante una aplicación en software.
- Obtener patrones de pérdida a partir de los números de secuencia del tráfico generado.
- Desarrollar un conjunto de scripts en R para caracterizar el comportamiento temporal de la pérdida de paquetes en tráfico de video streaming.

CAPÍTULO II. REDES INFORMÁTICAS

2.1 Introducción

Los últimos siglos estuvieron dominados por invenciones tecnológicas que aceleraron considerablemente la evolución de la humanidad. El siglo XVIII, fue la época de los grandes sistemas mecánicos que acompañaron a la Revolución Industrial. El siglo XIX fue la época de la máquina de vapor. Durante el siglo XX, la clave fue la obtención, el procesamiento y la distribución de información. Entre otras novedades, aparecieron las primeras instalaciones de redes telefónicas en todo el mundo, la invención de la radio y la televisión, el nacimiento y el crecimiento sin precedentes de la industria de la computación, el lanzamiento de los satélites de comunicación, y, por supuesto, la Internet.

Hay que recordar que las primeras computadoras eran caras y se encontraban aisladas. Sin embargo, después de unos años, ya que sus precios disminuyeron gradualmente, se comenzó a realizar los primeros experimentos para interconectarlas entre sí. En 1962, cuando el latente peligro de una guerra nuclear parecía inminente, surgía la interrogante de cómo controlar y distribuir la información en algún momento de crisis nuclear. Investigadores como Paul Baran, Donald Davies o Joseph Licklider publicaron las primeras ideas que describían los primeros pasos para la construcción de redes informáticas (Paul Baran and the Origins of the Internet).

Una red informática, es la agrupación de dos o más computadoras conectadas entre sí, en donde es posible que las personas pueden compartir archivos y periféricos tales como módems, impresoras, y unidades de CD-ROM (McMillan, 2015).

Las redes computacionales usualmente se clasifican con base a varios factores:

- Área geográfica
- Interconectividad
- Arquitectura

2.2 Clasificación de las redes por el área geográfica que cubren.

Dependiendo del área que cubre la red informática, se puede clasificar en:

- Personal Area Network (PAN)
- Local Area Network (LAN)
- Metropolitan Area Network (MAN)
- Wide Area Network (WAN)
- Internetworks

2.2.1 Personal area network (PAN)

Estas redes están diseñadas para la conexión de dispositivos de baja potencia situadas dentro de 1 m a 100 m de la otra (Hackmann, 2006).

Existen 3 enfoques diferentes de la tecnología PAN. El primero, Bluetooth, ofrece velocidades de datos de hasta 3 Mbps y distancias de hasta 100 m, con mucho menor consumo de energía que los dispositivos de 802.11b. Dado al bajo consumo de energía y la garantía de la interoperabilidad, Bluetooth cuenta con una gran aceptación en la comunidad móvil.

La segunda de estas tecnologías, 802.15.4, va más allá de Bluetooth en la velocidad de intercambio, llegando a ofrecer velocidades de hasta 250 kbps, y soportando fácilmente vínculos con un ciclo de trabajo muy bajo. Por lo tanto, es adecuado para los dispositivos que funcionan con baterías que deben sobrevivir hasta por un año entre cargas. 802.15.4 ya ha encontrado una amplia aceptación en la comunidad de redes de sensores (Hackmann, 2006).

Por último, banda de radios ultra ancha (UWB), emiten pulsos de baja potencia, pero con un gran ancho de banda que ofrecen tasas comparables a las de Ethernet. Sus altas tasas de datos y su relativamente bajo consumo de energía, lo hacen ideal para reemplazar los enlaces cableados cortos, como las que se encuentran entre los periféricos de la PC.

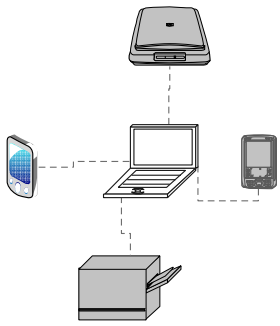


Imagen 2. 1 Red PAN

2.2.2 Local area network (LAN)

Si la red se encuentra dentro de un área relativamente pequeña, como un salón de clases, una escuela, o solo un edificio, se le conoce comúnmente como una red de área local (McMillan, 2015).

Debido a que los equipos en una LAN están relativamente cerca, su implementación no resulta costosa. Una LAN puede ser cableada, inalámbrica o en ambas formas a la vez. La mayoría de las LANs cableadas utilizan cables de cobre para transmitir, pero algunos ya utilizan fibra óptica. Por lo general, las redes LAN funcionan a velocidades de 100 Mbps a 10 Gbps, tienen un bajo retardo (microsegundos o nanosegundos), y muy pocos errores. La topología de muchas LANs cableadas suelen construirse a partir enlaces punto a punto. IEEE 802.3, popularmente llamada Ethernet, es el tipo más común de LANs cableadas (Wetherall & S.Tanenbaum, 2011).

Los nodos en una LAN están unidos entre sí con una cierta topología. Estas topologías incluyen:

- Bus
- Anillo
- Estrella

2.2.2.1 Redes LAN cableadas

Ya sabemos que una red de área local es una red informática que se ha diseñado para un área geográfica limitada, como un edificio o un campus. Aunque una LAN puede ser utilizada como una red aislada para conectar ordenadores en una organización con el único propósito de compartir recursos, la mayoría de las LANs hoy en día están también vinculadas a una red de área amplia (WAN) o Internet.

El término "cableado" se utiliza para diferenciar entre conexiones inalámbricas y las que implican cables. Una configuración de cableado utiliza cables físicos para transferir datos entre diferentes dispositivos y sistemas informáticos. En una pequeña red cableada, un router puede ser utilizado para conectar todos los equipos de la red. Las redes más grandes a menudo implican múltiples routers o switches que se conectan entre sí.

Han surgido varias tecnologías LANs, entre las principales están: Ethernet, Token Ring, Token Bus, FDDI y ATM LAN. Algunas de estas tecnologías sobrevivieron por un tiempo, pero Ethernet es, con mucho, la tecnología dominante (Wetherall & S.Tanenbaum, 2011).

2.2.2.2 Redes LAN Inalámbricas

La principal diferencia entre las redes cableadas e inalámbricas es que en las inalámbricas no se ven sujetas al uso de cables para establecer la comunicación entre dispositivos de la red, lo que genera una mejor movilidad.

Las redes inalámbricas hacen referencia al uso de señales infrarrojas o de radio frecuencia para compartir información y recursos entre los dispositivos. Actualmente hay una gran variedad de dispositivos inalámbricos, por ejemplo: teléfonos móviles, laptops, PDAs, sensores inalámbricos y receptores satelitales, entre otros.

El término Wi-Fi es la abreviatura de Wireless Fidelity, y es el nombre genérico con él que se le conocen a cualquier tipo de red 802.11, ya sea 802.11b, 802.11a, de doble banda, y así sucesivamente. El término se originó a partir de la Alianza Wi-Fi.

El estándar 802.11 se refiere a una familia de especificaciones desarrolladas por el IEEE para la tecnología de LAN inalámbrica. Más adelante se hablará con más detalles de los principales estándares inalámbricos (Ouellet, Padjen, Pfund, Fuller, & Blankenship, 2002).

2.2.3 Metropolitan area network (MAN)

A medida que la distancia entre las computadoras aumenta, una LAN se hace más difícil de instalar, y puede que se tenga que emplear medidas adicionales, tales como utilizar equipos de comunicaciones adicionales. Cuando la red se extiende por lo largo de una ciudad, puede ser referido como una red de área metropolitana (MAN). Gran parte de la misma tecnología, como los equipos de comunicaciones utilizados en redes LAN, se puede utilizar en las redes MANs. Aunque las velocidades alcanzadas en una red MAN suelen ser tan alta como en una LAN, se requiere de conexiones de alta velocidad, como la fibra óptica. El aumento de la distancia y los niveles de tecnología aumenta los costos de instalación y operación relativos de una red MAN (Poellabauer, 2014).

Una MAN trabaja en medio de una LAN y una WAN, ofreciendo un enlace ascendente de las redes LAN a WAN o Internet. Puede ser en forma de Ethernet, Token-ring, ATM, o Fiber Distributed Data Interface (FDDI).

Los ejemplos más conocidos de una red MAN son las redes de televisión por cable disponibles en muchas ciudades. Estos sistemas crecieron mediante un conjunto de antenas comunitarias anteriormente utilizadas en zonas con mala recepción de televisión a través del aire. Pero la televisión por cable no es el único ejemplo de una MAN, los avances recientes en la Internet inalámbrico de alta velocidad se han resultado en la creación de otra red MAN estandarizado como IEEE 802.16 y es conocido popularmente como WiMAX (Poellabauer, 2014).

2.2.4 Wide area network (WAN)

Una red WAN abarca un área más grande que una red MAN como un estado o un continente. Mientras que una LAN interconecta a los usuarios dentro de un edificio o en un campus, una WAN es capaz de conectar todas las redes LAN esparcidas en varios sitios, ya sea en la misma ciudad, en todo el país, o en todo el mundo (McMillan, 2015).

Las redes de telecomunicaciones son un ejemplo de redes WAN, estas redes proporcionan conectividad a redes WAN y LAN, puesto que están equipadas con enlaces troncales de muy alta velocidad.

2.2.5 Internetworks

Ocasionalmente un usuario en una red requiere comunicarse con otro usuario de una red distinta. En el mundo hay una gran variedad de redes existentes, a menudo con diferente hardware y software. Una colección de redes o una red de redes interconectadas genéricamente se les llama internetworks o internet, la más conocida es la Internet, que conecta las computadoras en todo el mundo. La Internet utiliza redes ISP para conectar las redes empresariales, redes domésticas, y muchas otras redes (Marsic, 2013).

En la Imagen 2.2, se aprecia un ejemplo de una internetwork.

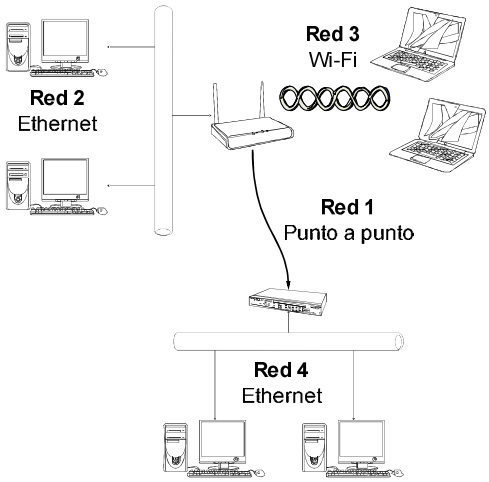


Imagen 2. 2 Intranetwork

Sabemos que una internetwork se forma cuando se interconectan redes distintas. Por ejemplo, la conexión de una LAN y una WAN o conectar dos LAN es la manera habitual para formar una internetwork, pero hay dos reglas generales que son útiles. En primer lugar, si las diferentes organizaciones han pagado para la construcción de diferentes zonas de la red y cada uno mantiene su propia zona, se tiene una internetwork en lugar de una sola red. En segundo lugar, si la tecnología de bajo nivel es diferente en las distintas zonas (por ejemplo, broadcast frente a point-to-point y alámbrico frente a inalámbrica), es probable que se trate de una internetwork.

Profundizando más en las internetworks, tenemos que hablar de cómo se pueden conectar dos redes diferentes. El dispositivo capaz de realizar esto, se le llama Gateway. Los Gateways hacen la conexión entre dos o más redes y proporcionan la traducción necesaria, tanto en términos de hardware y software. Se distinguen por la capa en la que trabajan en la jerarquía de protocolos (Marsic, 2013).

2.2.5.1 Jerarquía de protocolos

Para reducir la complejidad del diseño, la mayoría de las redes se organizan en una pila de capas o niveles, cada uno construido sobre la de abajo. El número, el nombre, el contenido y la función de cada capa difieren de una red a otra. El propósito de cada capa es ofrecer ciertos servicios a las capas superiores.

La idea fundamental es que una determinada pieza de software (o hardware) ofrezca un servicio a sus usuarios, pero manteniendo los detalles de su estado interno y algoritmos ocultos de ellos. Cuando la capa n en una máquina lleva a cabo una conversación con la capa n en otra máquina, las reglas y convenciones que se utilizan en esta conversación se conocen colectivamente como el protocolo de capa n. Básicamente, un protocolo es un acuerdo entre las partes que se comunican sobre cómo se debe proceder (Wetherall & S.Tanenbaum, 2011).

2.2.5.2 La relación entre los Servicios y los Protocolos

Un servicio y un protocolo son dos cosas distintas. Un servicio es un conjunto de operaciones que proporciona una capa a la capa superior de ella. El servicio le define al usuario qué operaciones de la capa están preparadas para llevar a cabo, pero no le dice en absoluto sobre cómo se implementan estas operaciones. Un servicio se refiere a una interfaz entre dos capas, siendo la capa inferior del proveedor de servicios y la capa superior que es el usuario del servicio.

Un protocolo, por el contrario, es un conjunto de normas que regulan el formato y el significado de los paquetes o mensajes que se intercambian entre las entidades pares dentro de una capa. De esta manera, el servicio y el protocolo están completamente separados (Wetherall & S.Tanenbaum, 2011).

2.3 Clasificación de las redes por su interconectividad

Los equipos en una red pueden conectarse entre sí de maneras diferentes. Por conexión se desea decir, ya sea de forma lógica, física, o de ambas maneras.

Las topologías son adecuadas para tareas específicas y tienen sus propias ventajas y desventajas. La elección de una depende del tipo y número de equipo que se utiliza, aplicaciones y tasa de transmisión de datos necesaria, tiempo de respuesta, y el coste previstos.

Una topología de red física hace énfasis en el hardware asociado con el sistema de los equipos, terminales remotos, servidores, y el cableado asociado entre los activos. Las más comunes son bus, estrella, árbol y anillo. Los híbridos de estos son, bus-estrella y anillo-estrella

Las topologías lógicas se enfocan en la representación del flujo de datos entre los nodos. La topografía lógica de una red se puede volver a configurar dinámicamente cuando se cuentan con equipo de red, tales como routers (Meador, 2008)

2.3.1 Topología de tipo bus

En una topología de tipo bus, todos los equipos se unen a la red a través de un solo cable, generalmente llamado Backbone, como se observa en la Imagen 2.3. Cuando se produce la comunicación entre nodos, el emisor transmite el mensaje a todos los nodos en la red, pero sólo el destinatario deseado acepta el mensaje.

Es fácil de implementar y al requerir menos cableado, la hace bajo costo, pero un fallo en el cable impedirá que todos los equipos tengan acceso a la red. Ya que funciona en modo broadcast, solo se permite que un equipo transmita datos a la vez y todas las demás solo podrán escuchar a la misma (Kharagpur, 2009).

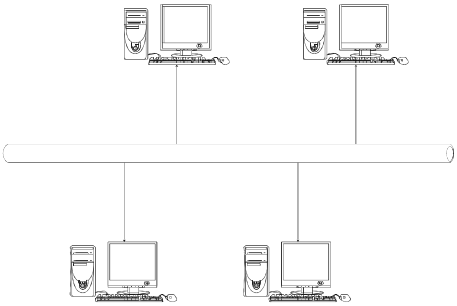


Imagen 2. 3 Topología de tipo bus

2.3.2 Topología de tipo estrella

En una topología de red estrella todos los nodos están conectados punto a punto directamente a un nodo central. Este nodo central puede ser una computadora, o un simple switch o hub. Los mensajes recibidos por el nodo central pueden ser transmitidos a todos los nodos subordinados, o si el dispositivo es de alta fidelidad, enviarlos sólo al nodo deseado.

Son relativamente fáciles de instalar y administrar, pero pueden ocurrir cuellos de botella debido a que todos los datos deben pasar a través del nodo central, además de que cualquier fallo en este nodo, detendría cualquier comunicación en la red (Kharagpur, 2009).

En la Imagen 2.4 se puede observar una topología de tipo estrella.

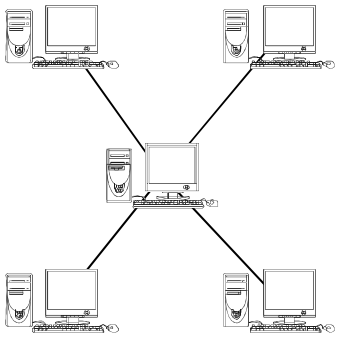


Imagen 2. 4 Topología de tipo estrella

2.3.3 Topología de tipo malla

Para permitir que cualquier host pueda enviar mensajes a cualquier otro host en la red, la solución más fácil es organizarlos en una topología de malla, con un enlace directo y dedicado entre cada par de hosts. Esta topología física se utiliza a veces, sobre todo cuando se requiere un alto rendimiento y alta redundancia para un pequeño número de equipos. Sin embargo, tiene dos grandes inconvenientes, requiere de más cableado, lo que lo hace más costoso que otras opciones y no es flexible, ya que tiene una limitada capacidad de expansión. Para añadir un nuevo nodo n enlaces tendrá que ser establecido debido a que el nuevo nodo tiene que estar conectado a cada uno de los nodos existentes a través de enlace dedicado (Meador, 2008).

En la Imagen 2.5 se puede observar una topología de tipo malla.

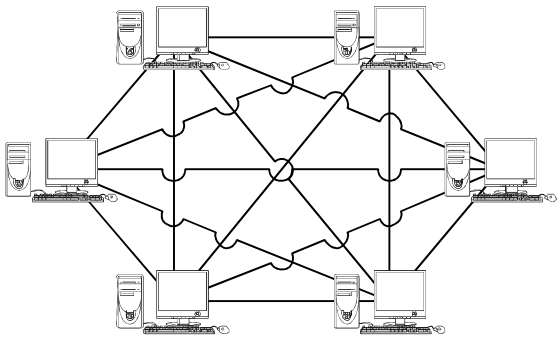


Imagen 2. 5 Topología de tipo malla

2.3.4 Topología de tipo anillo

Las topologías de tipo anillo, se caracterizan por tener cada nodo conectado a otros dos en la misma red. Al igual que en una red de tipo bus, cada host tiene una única interfaz física que se conecta al anillo, por lo que cualquier señal enviada por un host será recibida por todos los hosts conectados al anillo. En términos de redundancia, no es la mejor solución, ya que la señal sólo se desplaza en una sola dirección; por lo que si se corta uno de los elementos que componen el anillo, toda la red falla (Bonaventure, 2011).

En la Imagen 2.6 se muestra un ejemplo de una topología de tipo anillo.

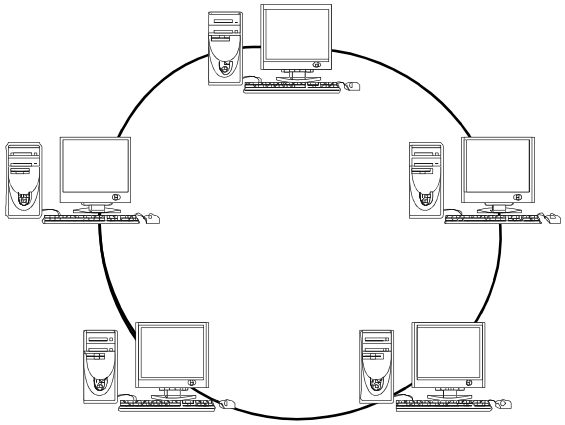


Imagen 2. 6 Topología de tipo anillo

2.3.5 Topología de tipo árbol

Esta topología puede ser considerada como una extensión de la topología de bus y estrella. Está construido a partir de un conjunto de topologías estrella conectadas a un nodo central, a través de una topología de tipo bus, distribuyendo así la funcionalidad del nodo central entre varios nodos de nivel superior.

Se conectan los equipos en forma de cascada, teniendo varios niveles organizados de nodos. Esto lo hace flexible y escalable, por ejemplo, se tiene una caja repetidora con 8 puertos, por lo que se tiene ocho hosts, esto se puede utilizar de manera normal, pero si es necesario agregar más hosts entonces se puede conectar dos o más repetidores en forma jerárquica (formato de árbol) y agregar más hosts (Meador, 2008).

En la Imagen 2.7 se puede observar una topología de tipo árbol.

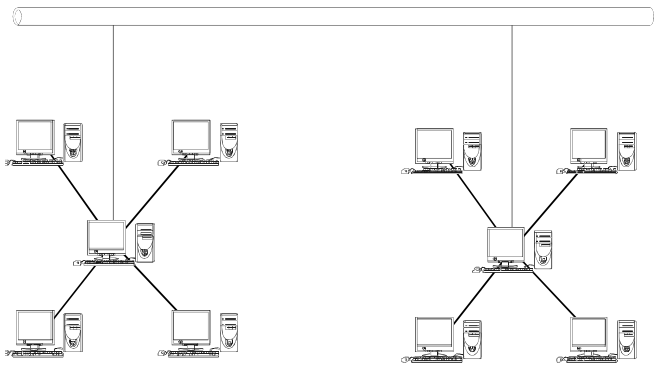


Imagen 2. 7 Topología de tipo árbol

2.4 Clasificación de las redes por su arquitectura.

Otra forma para clasificar las redes se basa en los roles que adquieren los equipos conectados en una red, y más específicamente en qué equipo tiene el control de las operaciones. Hay dos tipos básicos para esta clasificación, las redes peer-to-peer y redes basadas en servidor. La diferencia entre las dos está en torno a qué equipo está a cargo de la red. Una tercera clasificación, las redes basadas en el cliente, ha llegado a existir debido al aumento de las capacidades de una computadora cliente (Maly, 2003).

2.4.1 Redes peer-to-peer

Todos los equipos de una red peer-to-peer pueden considerarse iguales, es decir, ningún equipo está a cargo de la operación de la red. Cada computadora controla su propia información y es capaz de funcionar ya sea como un cliente o un servidor, dependiendo de lo que sea necesario en el momento.

Las redes peer-to-peer son baratas y fáciles de implementar, por lo que son populares en las redes domésticas y para el uso en pequeñas empresas. La mayoría de los sistemas operativos vienen incorporados con la capacidad de usar redes peer-to-peer.

El uso de redes peer-to-peer no implica ninguna medida de seguridad. Más bien, entre cada par se permite el acceso libre entre ellos para compartir sus recursos. De hecho, una red peer-to-peer se vuelve difícil de manejar cuando se añade más y más seguridad a los recursos. Esto se debe a los usuarios controlan su propia seguridad mediante el uso de contraseñas en cada acción que crean, tales como manejo de carpetas, impresoras y otros periféricos. Cada recurso compartido en realidad puede tener su propia contraseña (Maly, 2003).

2.4.2 Redes basadas en servidor

A diferencia de las redes peer-to-peer que operan sin un control central y son difíciles de proteger, una red basada en servidor ofrece un control centralizado y está diseñado para realizar operaciones seguras. Si bien todavía hay clientes y servidores en una red basada en el servidor, el servidor dedicado tiene control de la red.

La ventaja de hacer uso de estas redes es que tanto los archivos de datos y los servicios que serán utilizados por todos los usuarios se almacenan en un servidor. Esto ayudará a tener un punto central para establecer permisos en los accesos, y brindar seguridad a todos los datos en caso de que ocurra una pérdida (Maly, 2003).

2.5 Modelos de referencia

Un problema recurrente en la creación de red de redes es que la mayoría de las redes que se han desarrollado no siguen una misma implementación de hardware y software. Como resultado, las redes resultan ser incompatibles, volviéndose difícil de establecer una comunicación entre sí. Para solucionar este problema, se han creado modelos de red para ayudar a los diseñadores a implementar redes que pudieran comunicarse y trabajar en conjunto (interoperabilidad).

Existen dos modelos de red importantes: el modelo de referencia OSI y el modelo de referencia TCP/IP

2.5.1 Modelo de referencia OSI

El modelo de interconexión de sistemas abiertos (OSI, por sus siglas en inglés) es un esquema de red descriptivo de siete capas que especifica los requisitos para la comunicación entre dos ordenadores sin importar quién creó los protocolos y qué

equipo proveedor los apoya. Fue desarrollado por la Organización Internacional de Normalización (ISO, por sus siglas en inglés) en 1984.

El modelo OSI describe cómo la información o los datos recorren el camino desde los programas de aplicación (por ejemplo, editores de texto) a través de un medio de red (como alambre) hacia otro programa de aplicación que se encuentra en otra red.

Imagen 2. 8 Modelo OSI

En la Imagen 2.8 se observa las 7 capas numeradas del modelo OSI, cada una de las cuales describe una función de red específica que se debe realizar para que los paquetes de datos puedan viajar en la red desde el origen hasta el destino (Wetherall & S.Tanenbaum, 2011).

2.5.1.1 Capa de aplicación (capa 7)

La capa de aplicación es la más cercana al usuario y contiene los protocolos que son comúnmente necesarios para él, tales como HTTP (HyperText Transfer Protocol), que es la base para la World Wide Web y otros protocolos de aplicación se utilizan para la transferencia de archivos, correo electrónico y noticias de la red. Suministra servicios de red a las aplicaciones del usuario y a diferencia de las

otras capas, no proporciona servicios a ninguna otra capa OSI (Wetherall & S.Tanenbaum, 2011).

2.5.1.2 Capa de presentación (capa 6)

La capa de presentación, es responsable de otorgarle el formato correcto a los datos de una aplicación que serán enviados a la red.

A diferencia de las capas inferiores, que en su mayoría se preocupan más en cómo mover bits de un lado a otro, la capa de presentación se preocupa en la sintaxis y la semántica de la información transmitida. Con el fin de hacer posible la comunicación entre las computadoras con diferentes representaciones de datos internas (Wetherall & S.Tanenbaum, 2011).

2.5.1.3 Capa de sesión (capa 5)

La capa de sesión establece, administra y finaliza la comunicación entre los dos asociados. Esta capa decide el método de comunicación: half-duplex o full-duplex, incluyendo el control de diálogo (estar pendiente de a quien le toca transmitir), manejo de tokens (previene que las dos partes intenten una misma operación crítica al mismo tiempo), y sincronización del flujo de datos (Rigotti).

2.5.1.4 Capa de transporte (capa 4)

Aunque la capa de sesión es el responsable de decidir el método de comunicación, la capa de transporte es la que se encarga de establecer, mantener y terminar adecuadamente los circuitos virtuales necesarios para la comunicación. Brinda las funciones básicas para enviar datos al interlocutor, los cuales suelen incluir datos de multiplexación de diferentes aplicaciones, el establecimiento de integridad de los datos, y la gestión de los circuitos virtuales.

Dependiendo de la aplicación, la capa de transporte puede ofrecer comunicaciones seguras, orientadas a la conexión o a sin conexión, y comunicaciones de mejor esfuerzo (Wetherall & S.Tanenbaum, 2011).

2.5.1.5 Capa de red (capa 3)

Esta capa controla el funcionamiento de la subred y ofrece un sistema de direccionamiento lógico de extremo a extremo para que un paquete de datos se pueda enrutar a través de varias redes de capa 2 (Ethernet, Token Ring, Frame Relay, etc.). Esto nos permite enviar datos a cualquier computadora en el mundo, siempre y cuando haya una conexión de red física. Las rutas pueden estar basadas en tablas estáticas que rara vez cambian, o a menudo se pueden actualizar automáticamente para evitar componentes defectuosos. O pueden ser rutas dinámicas, que se determinan de nuevo para cada paquete dependiendo de la carga actual de la red. El dispositivo que nos permite lograr esto es el Router, a veces referido como un dispositivo de Capa 3 (Wetherall & S.Tanenbaum, 2011).

Traduce direcciones lógicas de red y nombres a su dirección física (por ejemplo, el nombre del dispositivo a su respectiva dirección MAC)

Resumiendo, la capa de red es responsable de:

- Direccionar
- Determinar las rutas para enviar datos
- Administrar problemas de red como la conmutación de paquetes, la congestión de datos y enrutamiento

2.5.1.6 Capa de enlace de datos (capa 2)

La capa de enlace de datos es la que conecta los protocolos de software con los protocolos de hardware. Es responsable de tomar los datos de las capas superiores y convertirlos a los bits necesarios para enviarlos a través del medio físico, y viceversa.

Al recibir los datos de la capa de red, crea tramas donde a cada trama se le agrega una dirección física de emisor/receptor (dirección MAC) en la cabecera para posteriormente pasarlo a la capa física (Simoneau, 2006).

Consta de 2 capas:

- Capa de Enlace Lógico (LLC): Define los métodos y proporciona información de direccionamiento para la comunicación entre los dispositivos de red.
- Control de acceso al medio (MAC): establece y mantiene vínculos entre los dispositivos que se comunican.

Resumiendo, la capa de enlace de datos proporciona las siguientes funciones:

- Permite el acceso un dispositivo a la red para enviar y recibir mensajes
- Ofrece un direccionamiento físico para que los datos de un dispositivo pueden ser enviados en la red.
- Proporciona la capacidad de detección de errores

2.5.1.7 Capa física (capa 1)

La capa física se encarga de la transmisión de bits puros en un canal de comunicación. Define la interfaz de conexión y las especificaciones, así como los requisitos del medio (cable), especificaciones eléctricas, mecánicas, funcionales y de procedimiento (Simoneau, 2006).

Funciones de la capa física:

- La transformación de los bits en señales
- Proporciona la sincronización de bits mediante un reloj.
- Administrar la forma en que un dispositivo se conecta a la red de comunicación.
- Definir la velocidad de transmisión.
- Definir la forma en la que los dispositivos están conectados al medio.
- Proporcionar topologías físicas

2.5.2 Modelo de referencia TCP/IP

Ahora pasemos al modelo de referencia utilizado en la mayoría de las redes informáticas actuales. Aunque es útil mencionar algunos aspectos claves, como ARPANET. La ARPANET era una red de investigación patrocinada por el Departamento de Defensa de los Estados Unidos. Eventualmente conectaron a cientos de universidades e instalaciones de gobierno, utilizando líneas telefónicas arrendadas. Cuando más tarde se agregaron las redes satelitales y de radio, los protocolos existentes tenían problemas para trabajar entre ellos, por lo que era necesaria una nueva arquitectura de referencia. Los primeros pasos para la creación de esta nueva arquitectura se dieron en 1974, cuando fue descrita por primera vez por Cerf y Kahn, más tarde perfeccionada y definida como un estándar en la comunidad de Internet (Braden, 1989). Llegando a ser conocida como el modelo de referencia TCP/IP, debido a sus dos protocolos principales (Cerf & Kahn, 1974).

El protocolo IP se definió en el RFC 760 (Request for Comments 760) en 1980; TCP en el RFC 793 en 1981. TCP/IP son un conjunto de protocolos relacionados que trabajan juntos para permitir la comunicación de las computadoras conectadas en la red. Algunos de los protocolos más comunes incluyen HTTP, protocolo de transferencia de archivos (FTP), SMTP, Protocolo de mensajes de control de Internet (ICMP), y el Protocolo de oficina de correos (POP). Cada uno de estos protocolos utiliza como su fundamento base el protocolo IP para el movimiento de datos en una red. El actual modelo de referencia TCP/IP abarca cuatro capas (capas 2-5). Estas junto con la capa física (capa 1) conforman el modelo de referencia de cinco capas TCP híbrido/IP (véase la Imagen 2.9)

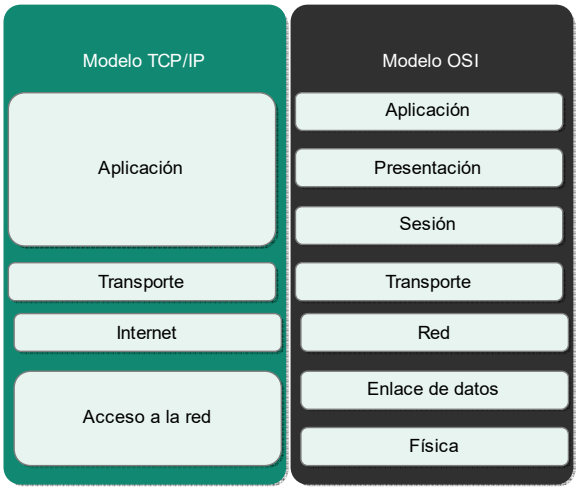


Imagen 2. 9 Comparativa modelo TCP/IP y modelo OSI

La capa 2 del modelo TCP IP a menudo se designa como la capa de enlace de datos, o también se le conoce como la capa de acceso a la red. Corresponde a las dos primeras capas del modelo ISO/OSI (capa física, capa de enlace de datos). La capa 3 del modelo TCP/IP (capa de Internet) también se le llama la capa de red o capa Internetwork y corresponde a la capa 3 del modelo ISO/OSI (capa de red).

La capa 4 del modelo TCP/IP (capa de transporte) corresponde a la capa 4 del modelo de referencia ISO/OSI (capa de transporte).

La capa 5 del modelo TCP/IP (capa de aplicación) corresponde a las capas 5 - 7 del modelo ISO/OSI (capa de sesión, capa de presentación, capa de aplicación).

En las secciones siguientes, se examinarán con más detalle cada una de las capas del modelo TCP/IP.

2.5.2.1 Capa de aplicación (capa 4)

A diferencia del modelo OSI, el modelo TCP/IP no cuenta con capa de sesión y presentación. En su lugar, las aplicaciones sólo incluyen las funciones de sesión y de presentación que requieran. La experiencia con el modelo OSI ha demostrado que estas capas son de poca utilidad para la mayoría de aplicaciones.

Funciona principalmente como una interfaz para los programas de aplicación que deseen comunicarse a través de la red (comunicación de proceso a proceso).

Los protocolos y servicios de la capa de aplicación normalmente llevan a cabo traducciones y transformaciones de datos entre los programas a nivel semántico. Entre los servicios que se ofrecen son: sistema de nombres de servicio (DNS), que traducen las direcciones IP en nombres legibles y viceversa; redirección de servicios, que desvían las peticiones que no pueden ser llenados a otro host, así como servicios de directorio y servicios de gestión de red (Wetherall & S.Tanenbaum, 2011).

2.5.2.2 Capa de transporte (capa 3)

La capa de transporte en el modelo TCP/IP corresponde aproximadamente a la capa 4 del modelo de referencia OSI. Su tarea principal implica el establecimiento y uso de una conexión entre dos programas de aplicaciones que residen en diferentes equipos de la red.

El protocolo de la capa de transporte establece una conexión directa virtual de extremo a extremo, para permitir que varios programas de aplicación en el mismo equipo tengan comunicación paralela. A cada programa se le asigna un número de puerto para proporcionar una identificación única, de esta forma cada unidad de datos enviados debe contener el número de puerto del emisor y el receptor con el fin de tener una transmisión correcta.

Se utiliza un complejo control de flujo para, en la mayor medida posible, asegurar que no se presenten situaciones de sobrecarga con el fin de evitar la congestión de

la red. Por último, se toman medidas para garantizar que los datos transmitidos lleguen libres de errores al receptor y en el orden correcto (usando números de secuencia). En el caso de que los paquetes resulten defectuosos. Un mecanismo de acuse de recibo se proporciona con el cual el receptor puede confirmar que los paquetes se han transmitido correctamente, o enviar una nueva solicitud.

Dos protocolos de transporte son definidos en esta capa, el primero es TCP (Transmission Control Protocol), un protocolo orientado a conexiones fiables que permite que flujos de datos sean entregados sin error al destinatario. El segundo protocolo es UDP (User Datagram Protocol), el cual es un protocolo sin conexión fiable para las aplicaciones que no quieren secuencias TCP o control de flujo. Es ampliamente utilizado para aplicaciones en las que la entrega rápida es más importante que la entrega precisa, como la transmisión de voz o vídeo (Wetherall & S.Tanenbaum, 2011).

2.5.2.3 Capa de Internet (capa 2)

La función de la capa de Internet es permitir que los hosts transmitan paquetes que viajen de manera independiente al destino (potencialmente en diferentes redes). Incluso pueden llegar en un orden completamente diferente de lo que fueron enviados, en cuyo caso es el trabajo de las capas superiores para reorganizarlas, si es que se desea un orden en la entrega.

La capa de Internet define un formato oficial de paquete y un protocolo llamado IP (Protocolo de Internet), además de ICMP (Internet Control Message Protocol) que le ayuda en la función. Como se mencionó, el trabajo de la capa es entregar los paquetes IP a donde deben ir. El enrutamiento de paquetes es claramente un problema importante aquí, así como la congestión (Wetherall & S.Tanenbaum, 2011).

2.5.2.4 Capa de acceso a la red (capa 1)

También llamada capa de enlace o capa de enlace de datos, es la interfaz con el hardware de la red. Esta interfaz puede o no puede proporcionar entrega confiable, y puede ser de paquetes u orientada a flujos. De hecho, TCP/IP no especifica ningún protocolo aquí, pero se puede utilizar casi cualquier interfaz de red disponible.

La tarea principal de la capa es la transmisión segura de los paquetes individuales entre dos sistemas finales adyacentes. Las secuencias de bits a transmitir se agrupan en unidades fijas y se les proporcionará la información adicional necesaria para la transmisión, por ejemplo, sumas de control para la detección de errores. Sobre esta capa se hace una distinción entre los servicios garantizados y no garantizados. Con los servicios de garantía, se eliminan los paquetes de datos reconocidos como defectuosos. Posteriormente se lleva a cabo una solicitud para la re-transmisión, pero en una capa superior. En contraste, un servicio garantizado se hace cargo de la solicitud de una retransmisión en sí.

Los protocolos más importantes de la capa de acceso a la red del modelo de referencia TCP/IP son los de la IEEE, basado en el estándar de LAN IEEE 802. Estas son tecnologías como Ethernet (IEEE 802.3), Token Ring y FDDI (IEEE 802.5), así como diferentes tecnologías WLAN inalámbrica (IEEE 802.11) (Meinel & Sack, 2013).

2.6 Redes Inalámbricas

Actualmente, Wi-Fi es una tecnología de red inalámbrica muy popular. Hay más de cientos de millones de dispositivos Wi-Fi en todo el mundo. El desarrollo de esta tecnología comenzó a finales de 1980 con la red inalámbrica patentada WaveLAN. A principios de 1990, el IEEE creó el grupo de trabajo 802.11 para estandarizar una familia de tecnologías inalámbricas. Este grupo fue muy prolífico y produjo varios estándares de redes inalámbricas que utilizan diferentes rangos de frecuencia y diferentes capas físicas, véase Tabla 2.1.

Las redes 802.11 utilizan la técnica CSMA/CA Medium Access Control y todos asumen la misma arquitectura y utilizan el mismo formato de trama.

El modo más popular de utilizar las redes 802.11 es la de infraestructura. Consiste en conectar a los clientes, tales como computadoras, portátiles y teléfonos inteligentes, a otra red, tal como una intranet de la empresa o la Internet. Cada cliente se asocia con un AP (Access Point) que a su vez está conectado a la red. El cliente envía y recibe sus paquetes a través del AP.

El otro modo es una red ad hoc. Este es un conjunto de equipos que están asociados de modo que puedan enviar directamente tramas el uno al otro. No hay punto de acceso. Dado que el acceso a Internet es la aplicación de la tecnología inalámbrica, redes ad hoc no son muy populares (Bonaventure, 2011).

Estándar	Frecuencia	Velocidad Máxima	Rango (m) Interior/Exterior
802.11	2.4 GHz	2 Mbps	20/100
802.11a	5 GHz	54 Mbps	35/120
802.11b	2.4 GHz	11 Mbps	38/140
802.11g	2.4 GHz	54 Mbps	38/140
802.11n	2.4/5 GHz	150 Mbps	70/250

Tabla 2. 1 Estándares 802.11

2.6.1 Arquitecturas

La topología de una red inalámbrica es dinámica; por lo tanto, la dirección de destino no siempre corresponde a la de ubicación. Este destino plantea un problema al reenviar tramas a través de la red al destino previsto. El estándar IEEE 802.11, define tres tipos de servicios: el conjunto de servicio básico (BSS), el conjunto de servicio extendido (ESS) y el conjunto de servicio básico independiente (IBSS)

2.6.1.1 Conjunto de servicio básico (BSS)

El BSS consiste en al menos un punto de acceso conectado a la infraestructura de red cableada y un conjunto de terminales inalámbricas (véase la Imagen 2.10). Las configuraciones BSS se basan en un punto de acceso que actúa como servidor para una sola red LAN inalámbrica o canal. Las comunicaciones entre dos puntos finales fluyen desde una terminal al punto de acceso y del punto de acceso a la otra terminal (Ouellet, Padjen, Pfund, Fuller, & Blankenship, 2002).

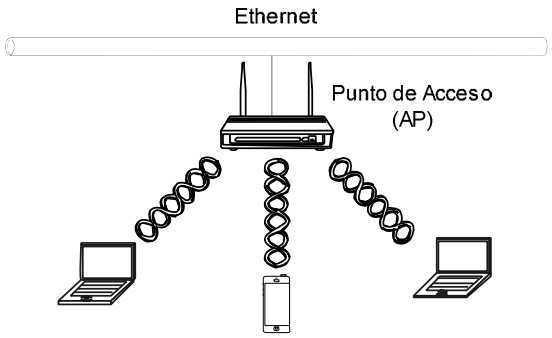


Imagen 2. 10 Arquitectura BSS

2.6.1.2 Conjunto de servicio extendido (ESS)

Las topologías ESS consisten en una serie de conjuntos BSS (cada uno contiene un AP), comúnmente conocida como celdas. Estas suelen estar conectados entre sí por algún sistema de distribución, por ejemplo, Ethernet, (ver Imagen 2.11). Las terminales móviles pueden moverse entre los puntos de acceso, lo que hace posible una cobertura perfecta en toda la ESS (Ouellet, Padjen, Pfund, Fuller, & Blankenship, 2002).

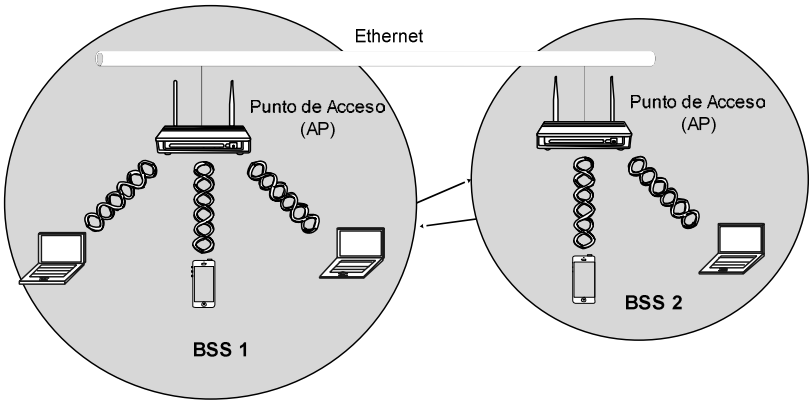


Imagen 2. 11 Arquitectura ESS

2.6.1.3 Conjunto de servicio básico independiente (IBSS)

El conjunto IBSS también son conocidos como una red de configuración o ad-hoc independiente. Lógicamente, una configuración IBSS es muy similar a una red doméstica o de oficina peer-to-peer en la que no se requiere de un servidor para funcionar (ver Imagen 2.12). Esta topología se compone de un número de estaciones inalámbricas que se comunican directamente uno con el otro, sin la intervención de un punto de acceso o cualquier conexión de red cableada. En general, las implementaciones ad-hoc cubren una pequeña zona (limitada) y no están conectados a una red (Ouellet, Padjen, Pfund, Fuller, & Blankenship, 2002).

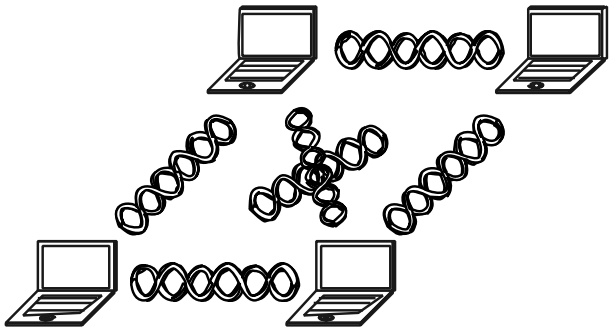


Imagen 2. 12 Arquitectura IBSS

2.6.2 Estándares 802.11

El estándar 802.11 se refiere a una familia de especificaciones desarrolladas por el IEEE para la tecnología LAN inalámbrica. La norma 802.11 especifica una interfaz sobre el aire entre un cliente inalámbrico y una estación base o entre dos clientes inalámbricos.

2.6.2.1 Estándar 802.11a

802.11a utiliza una técnica de codificación llamada Orthogonal Frequency Division Multiplexing (OFDM). Los equipos 802.11a funcionarán bajo la frecuencia de 5 GHz y soporta hasta una taza de 54 Mbps de datos. Además de la frecuencia y la asignación de ancho de banda, un parámetro clave que está regulada por las diferentes autoridades es la potencia de salida de transmisión admisible. La potencia de salida está directamente relacionada con la gama de cobertura que una radio particular puede alcanzar. El espectro de 5 GHz se divide en tres de "dominios". La primera 100 MHz en la sección inferior se limita a una potencia de salida máxima de 50 mW. El segundo de 100 MHz tiene un más generoso presupuesto de potencia de 250 mW, mientras que la parte superior 100 MHz se delega para aplicaciones en exteriores, con un máximo de potencia de 1W (Ouellet, Padjen, Pfund, Fuller, & Blankenship, 2002).

2.6.2.2 Estándar 802.11b

El 16 de septiembre de 1999, el IEEE validó una revisión del estándar 802.11, llamada 802.11-Tasa alta (HR/DSSS) o 802.11b, que ofrece velocidades de datos mucho más altas, manteniendo al mismo tiempo los protocolos 802.11. La arquitectura básica, características y servicios de 802.11b están definidos en el estándar original 802.11 ya que la especificación revisada sólo afecta a la capa física, añadiendo mayores velocidades de datos y conectividad más robusta.

Ofrece 11 Mbps de transmisión con tazas de 5,5, 2 y 1 Mbps en la banda de 2,4 GHz. 802.11b utiliza sólo el espectro extendido de secuencia directa (DSSS). Utiliza un desplazamiento de frecuencia dinámica, lo que permite que las velocidades de datos se ajusten automáticamente para compensar la interferencia o problemas en el canal de radio (Wetherall & S.Tanenbaum, 2011).

2.6.2.3 Estándar 802.11g

El estándar 802.11g está diseñado como sucesor del 802.11b. El estándar 802.11g proporciona velocidades de datos opcionales de hasta 54 Mbps, y requiere la compatibilidad con los dispositivos 802.11b para proteger las inversiones en instalaciones WLAN con 802.11b. Por lo que un punto de acceso 802.11g soporta clientes 802.11b y 802.11g. Del mismo modo, un ordenador portátil con una tarjeta 802.11g será capaz de acceder a los puntos de acceso 802.11b existentes, así como a los nuevos puntos de acceso 802.11g. Esto se debe a que las redes LAN inalámbricas basadas en 802.11g utilizan la misma banda de 2,4 GHz que 802.11b usa (Marsic, 2013).

2.6.2.4 Estándar 802.11n

Es un nuevo estándar 802.11 que promete ser superior a todas sus antecesoras, pudiendo obtener hasta 5 veces más velocidad que ellas, con una tasa de transferencia de al menos 100 Mbps.

Las modificaciones del 802.11n incluyen muchos ajustes que mejoran el rango, la fiabilidad y rendimiento de las WLANs. En la capa física se han añadido técnicas de procesamiento de señal y modulación avanzadas para explotar múltiples antenas y canales más anchos. En la capa de control de acceso al medio (MAC), las extensiones de protocolos hacen un uso más eficiente del ancho de banda disponible.

Las tasas de datos en 802.11n son significativamente mejores sobre las conseguidas por 802.11a y 802.11g, fundamentalmente por el uso de las múltiples antenas, o MIMO (Multiple-Input, Multiple-Output), y el uso de canales de 40 MHz. Juntos, estas mejoras pueden aumentar las velocidades de datos de hasta 600 Mbps. Al implementar 802.11n, es críticamente importante entender lo que estas mejoras de alto rendimiento realmente hacen, cómo impactan y coexisten con otras tecnologías WLAN (Meinel & Sack, 2013).

2.6.2.4.1 MIMO

Una de las mejoras 802.11n que permite la capacidad de recibir y/o transmitir simultáneamente a través de múltiples antenas. 802.11n define muchas "M x N" configuraciones de antena, que van desde "1 x 1" hasta "4 x 4". Esto se refiere al número de transmisión (M) y de recepción (N) antenas, por ejemplo, un punto de acceso con dos de transmisión y tres antenas de recepción es un dispositivo MIMO "2 x 3" (véase Imagen 2.13).

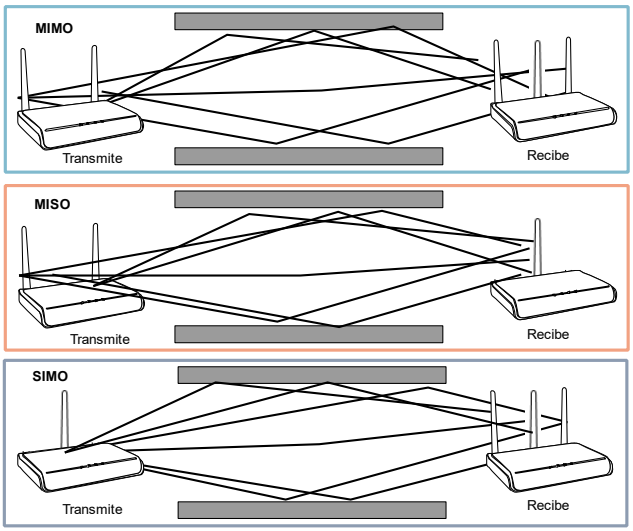


Imagen 2. 13 Tecnología MIMO

En general, cuantas más antenas se utilizan al mismo tiempo, mayor será la velocidad máxima de datos. Sin embargo, varias antenas por si mismas no aumentan la velocidad de datos o rango. Esas mejoras provienen de cómo el dispositivo MIMO utiliza las técnicas avanzadas de procesamiento de señal introducidas por el estándar 802.11n.

La clave asociada a MIMO es 'SDM' (multiplexación por división espacial). SDM proporciona un sistema MIMO con una transmisión de datos superior. Permite la transmisión de múltiples flujos de datos, lo que da un mayor rendimiento debido a las múltiples cadenas de antenas. Mientras que SDM es una técnica de multiplexación para aumentar la velocidad de datos en general, la codificación espacio-temporal por bloques (STBC) y la combinación de relación máxima (MRC) son las técnicas que mejoran la relación señal-ruido y la relación de interferencia, SNR o SINR. Estas técnicas se pueden combinar en algunas condiciones (Whitepaper 802.11n Primer, 2008).

2.6.2.4.2 Multiplexación por división espacial (SDM)

Para lograr un mayor rendimiento utilizando múltiples antenas, la técnica clave utilizada es la multiplexación por división espacial. Como su nombre lo indica, esta

técnica implica la multiplexación de múltiples flujos de datos a través de las dimensiones espaciales (es decir, antenas de transmisión separadas por ubicación o espacio). Con SDM, múltiples antenas de transmisión, apropiadamente espaciados, se utilizan para transmitir flujos de datos independientes, que individualmente pueden ser recuperados en el receptor. La Imagen 2.14 ilustra el funcionamiento básico de SDM con MIMO.

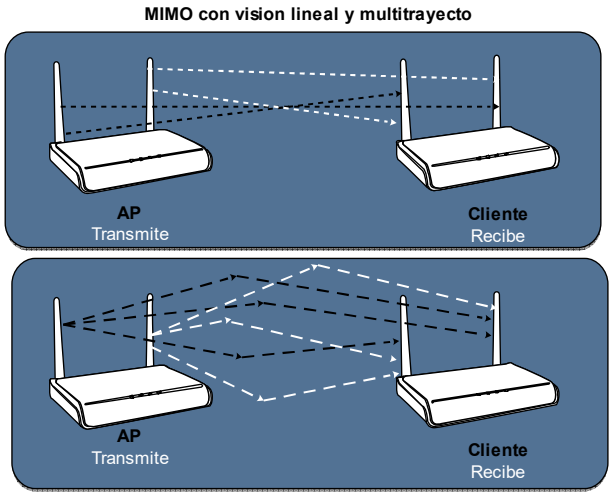


Imagen 2. 14 Multiplexación por división espacial

Debido a que cada transmisión se propaga a lo largo de un camino diferente, esas piezas - llamados flujos espaciales - llegan con diferente fuerza y retrasos. El receptor con SDM activo es capaz de volver a rearmar el flujo de la señal. Dos flujos espaciales multiplexado en un solo canal (frecuencia de radio) permiten duplicar la capacidad y por lo tanto maximiza la velocidad de datos. Todos los puntos de acceso 802.11n deben implementar al menos dos flujos espaciales, hasta un máximo de cuatro (Pau & Ogunfunmi, 2008).

2.6.2.4.3 Codificación espacio-temporal por bloques (STBC)

La codificación espacio-temporal por bloques se utiliza para lograr la diversidad espacial utilizando múltiples antenas de transmisión. Con STBC se envía flujo saliente de señal de forma redundante, utilizando hasta cuatro flujos espaciales codificados de manera diferente, cada una transmitida a través de una antena diferente, véase Imagen 2.15.

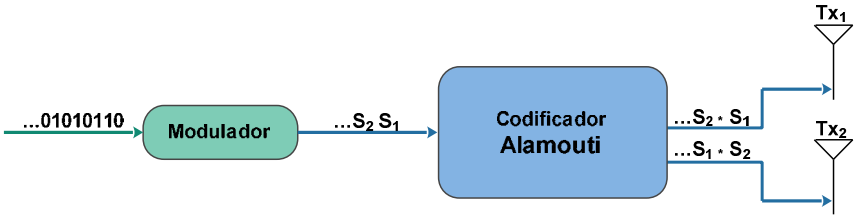


Imagen 2. 15 Codificación espacio-temporal por bloques (STBC)

Mediante la comparación de las secuencias espaciales, el receptor tiene una mejor oportunidad de determinar con precisión el flujo de señal original en casos de interferencia de radio frecuencia y distorsión. Es decir, STBC mejora la fiabilidad mediante la reducción de la tasa de error experimentado en la relación señal/ruido (SNR). Esta característica opcional se puede combinar junto con SDM, pero sólo cuando el número de antenas de transmisión supera las de recepción.

La forma más común de codificación STBC, es la codificación Alamouti. Esta propaga una secuencia espacial a más de dos flujos de tiempo-espacio, tomando un par de bits de flujo de datos y realizando operaciones sobre ellos en intervalos de tiempo consecutivos. Al procesar esta secuencia de dos flujos de espacio-tiempo, el receptor es capaz de reconstituir el flujo original, incluso en presencia de ruido y distorsión en el canal (Pau & Ogunfunmi, 2008).

2.6.2.4.4 Combinación de relación máxima (MRC)

Es una técnica que utiliza múltiples antenas de recepción para extraer una mejor señal: se requiere más de una cadena de antenas de recepción, pero se puede trabajar con una única antena de transmisión, véase Imagen 2.16.

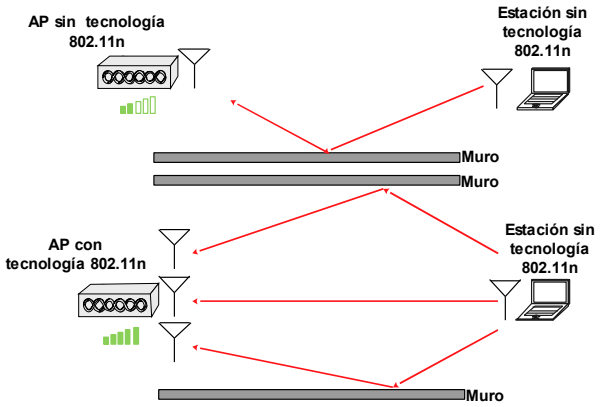


Imagen 2. 16 Combinación de relación máxima (MRC)

Puesto que el equipo 802.11n ya cuenta con múltiples cadenas de recepción independientes, MRC se desenvuelve con facilidad en los productos 802.11n.

MRC permite al receptor procesar las dos señales de forma independiente, y luego combinarlas, ponderado por la intensidad de cada señal. En términos sencillos, si una antena se encuentra en 'nulo' y tiene una mala señal, mientras que el otro es probable que tenga una buena señal, es posible recibir el flujo de datos. El receptor sólo sufrirá cuando simultáneamente todas las antenas lleguen a tener mala señal.

MRC no está cubierta explícitamente en el estándar 802.11n, ya que puede ser implementado sólo en el receptor, sin cambios en el extremo de transmisión. Sin embargo, en la actualidad, la mayoría de los proveedores de chips 802.11n aplican una forma de diversidad de recepción como MRC (Aruba Networks).

CAPÍTULO III. CALIDAD DE SERVICIO (QoS)

En la actualidad ya es algo básico que las organizaciones y empresas hagan uso de al menos un tipo de red. Estas redes suelen transportar el tráfico de un gran número de aplicaciones, incluyendo aplicaciones de voz en tiempo real, transmisión de vídeo de alta calidad y flujos de información sensibles a los retardos. Las redes deben ofrecer servicios fiables, medibles, y en ocasiones garantizados por la gestión de los parámetros de ancho de banda, retardo, jitter y pérdida de paquetes en una red.

Calidad de servicio es el término utilizado para definir la capacidad de una red para proporcionar diferentes niveles de garantías de servicio para las diversas formas de tráfico. Permite a los administradores de red asignar cierta prioridad de tráfico sobre los demás o los niveles reales de calidad con respecto al ancho de banda de red o retardo de extremo a extremo (Flannagan, Durand, Sommerville, Buchmann, & Fuller, 2001).

El uso de servicios de QoS en una red, podría beneficiar tanto el rendimiento, como la calidad de la red, esto mediante algunas funciones como son:

- Asignar ancho de banda dedicado
- Mejorar las características de pérdida
- Evitar y controlar la congestión de red
- Organizar el tráfico de la red
- Establecer prioridades de tráfico en toda la red

Pero para comenzar, lo primero es definir claramente los objetivos de la organización. Por ejemplo, una de las primeras preguntas que surgen durante una implementación de QoS son: ¿Cuántas clases de tráfico se establecerán en la red? ¿Y qué uso tendrán cada uno?

Para ayudar a abordar estas cuestiones, Cisco ha adoptado una nueva iniciativa llamada "QoS Baseline." Es un documento diseñado por expertos en calidad de

servicio dentro de Cisco, para unificar los servicios QoS en la empresa. También proporciona recomendaciones, basado en estándares para ayudar a asegurar que el diseño e implementaciones de la calidad de servicio sean coherentes. Este modelo define hasta 11 clases de tráfico que pueden ser vistos como críticos para una empresa. Un resumen de estas clases y sus respectivas recomendaciones se presentan en la Tabla 3.1 (Cisco).

Aplicación	Clasificación L3			L2
	IPP	PHB	DSCP	CoS
Enrutamiento	6	CS5	48	6
Voz	5	EF	46	5
Video conferencia	4	AF41	34	4
Transmisión de video	4	CS4	32	4
Datos de misión critica	3	AF31	26	3
Señalización de llamadas	3	CS3	24	3
Datos de transacciones	2	AF21	18	2
Administración de red	2	CS2	16	2
Flujo masivo de datos	1	AF11	10	1
Basurero	1	CS1	8	1
Mejor esfuerzo	0	0	0	0

Tabla 3. 1 Cisco QoS Baseline

En este capítulo se describen las principales características de los elementos más relevantes que participan a la hora de brindar una mejor calidad de servicio.

3.1 Niveles de QoS

Los sistemas de calidad se pueden clasificar en tres diferentes niveles, también conocidos como modelos de servicio. Cada uno describe un conjunto de características de calidad de servicio de extremo a extremo. Los tres servicios son:

- Servicio de mejor esfuerzo.

- Servicio integrado.
- Servicio diferenciado.

Los modelos de servicio se diferencian entre sí en la forma en que permiten a las aplicaciones enviar datos y en las formas en que la red debe entregar los datos. Por ejemplo, un modelo de servicio diferenciado se aplica a las aplicaciones en tiempo real, tales como conferencias de audio, vídeo y telefonía IP. Hay que tener en cuenta los siguientes factores a la hora de decidir qué tipo de servicio desplegar en la red (Flannagan, Durand, Sommerville, Buchmann, & Fuller, 2001):

- La aplicación o el problema que están tratando de resolver
- La clase de habilidad que desea asignar a sus recursos
- Análisis costo-beneficio. Por ejemplo, el costo de la implementación y el despliegue de un servicio diferenciado resultará más caro que implementar un servicio de mejor esfuerzo

3.1.1 Servicio de mejor esfuerzo

Como su nombre lo indica, es cuando la red hará todo intento posible de entregar un paquete a su destino. En este modelo de servicio, una aplicación envía datos cada vez que se requiera, en cualquier cantidad, y sin notificar a la red, por lo que la garantía de que llegue a su destino es nula.

Este servicio es adecuado para una amplia gama de aplicaciones en red, tales como la transferencia de archivos en general o aplicaciones de correo electrónico. Sin embargo, no es un modelo óptimo para aplicaciones que son sensibles a los retrasos en la red, las fluctuaciones de ancho de banda, y otras condiciones inconsistentes de red (Cisco, 2014).

Para que una red de mejor esfuerzo funcione bien, es necesario que las aplicaciones entiendan la variabilidad dinámica de la red y que deben adaptarse a esa variabilidad, no que la red se adapte a ellos.

3.1.2 Servicio integrado

El modelo de servicio integrado proporciona a las aplicaciones un nivel de servicio garantizado mediante la negociación de parámetros de red. Las aplicaciones solicitan el nivel de servicio necesario para funcionar correctamente y se basan en el mecanismo de QoS para reservar los recursos necesarios de red antes de que comiencen a enviar datos. Hay que tener en cuenta que las aplicaciones no enviarán el tráfico hasta que reciban una señal que indique que la red está capacitada para manejar la carga y proporcionar la calidad de servicio extremo a extremo.

La solicitud se realiza mediante señalización explícita; la aplicación informa a la red su perfil de tráfico y solicita un tipo particular de servicio que abarque sus requisitos de ancho de banda y de retardo. La aplicación espera la confirmación de la red para luego comenzar a transmitir.

La red se compromete a cumplir con los requisitos de calidad de servicio de la aplicación, siempre y cuando el tráfico se mantenga dentro de las especificaciones del perfil. Para lograr esto, la red utiliza un proceso llamado control de admisión. El control de admisión es el mecanismo que impide que la red quede sobrecargada.

Una vez que la aplicación inicia la transmisión de datos, los recursos de red reservados para la aplicación se mantienen hasta que la aplicación termine o hasta que la reserva de ancho de banda sea superior a lo solicitado por la aplicación. La red llevará a cabo tareas de mantenimiento del estado de flujo, clasificación, vigilancia y gestión de colas inteligentes de paquetes para satisfacer la calidad de servicio requerida (Flannagan, Durand, Sommerville, Buchmann, & Fuller, 2001).

3.1.3 Servicio diferenciado

Es un modelo de servicios múltiples que pueden satisfacer diferentes requisitos de QoS. Sin embargo, a diferencia del modelo de servicio integrado, una aplicación que utiliza un servicio diferenciado no señaliza explícitamente al router antes de enviar datos.

Para un servicio diferenciado, la red intenta entregar un tipo particular de servicio en base a la calidad de servicio especificado en cada paquete. Esta especificación puede ocurrir de diferentes maneras, por ejemplo, utilizando la configuración de bits de prioridad IP en paquetes IP o direcciones de origen y de destino. La red utiliza la especificación de QoS para clasificar, marcar y controlar el tráfico. Este modelo se utiliza para aplicaciones de prioridad crítica y para proporcionar calidad de servicio extremo a extremo. Por lo general, es apropiado para los flujos agregados, ya que realiza un nivel de clasificación del tráfico relativamente basto.

El tráfico de red se puede clasificar por direcciones de red, protocolos y puertos, interfaces de entrada o cualquiera que sea la clasificación que se pueden lograr mediante el uso de una lista de acceso estándar o extendido.

La premisa del servicio diferenciado es que los routers en la red central, manejan los paquetes de diferentes flujos de tráfico desviándolas usando un comportamiento por salto diferente (PHB) (Bai, Atiquzzaman, & Ivancic, 2002).

3.2 Parámetros QoS

Medir con precisión los niveles de QoS en una red se ha vuelto uno de los puntos importantes para tener una red robusta. Las mediciones se pueden utilizar para analizar casos problemáticos y mejorar el rendimiento (optimizar) de la red o servicios de manera eficaz y rentable. Que a su vez ayudará a mantener un alto grado de satisfacción con los usuarios finales y dar una ventaja competitiva.

Lo que el usuario final percibe se analiza principalmente en términos de la integridad del servicio, que se refiere al rendimiento, el retardo (y la variación del retardo o jitter) y la pérdida de datos de un servicio durante la comunicación. Tales parámetros de rendimiento pueden ser obtenidos mediante los mensajes de señalización en diagramas de flujo disponibles en los estándares o mediante la supervisión directa de los mensajes intercambiados entre los dominios de la red a través de las interfaces correspondientes.

Los parámetros QoS son características particularmente relevantes para ciertas clases de tráfico que afectan a la calidad de un servicio, incluso si es sólo una transferencia de datos o una transmisión en tiempo real. Los parámetros básicos para dar una buena calidad de servicio son (Soldani, Li, & Cuny, 2006):

- Pérdida de paquetes
- Retardo extremo a extremo
- Jitter

3.2.1 Pérdida de Paquetes

La fiabilidad es una característica que un flujo de datos requiere. La falta de ella, significa que uno o más paquetes se han perdido en el proceso de transmisión, lo que implica retransmitirlos de nuevo. Las redes inalámbricas e IP no pueden proporcionar una garantía de que los paquetes serán entregados en su totalidad, y que no se perderán (descartarán) algunos paquetes, si al llegar, sus buffers ya están llenos. Esta pérdida de paquetes puede ser causada por otros factores como la

degradación de la señal, las altas cargas en los enlaces de red, los paquetes que están dañados o defectos en los elementos de red.

Las redes inalámbricas tienen una mayor probabilidad de pérdida dado a las múltiples interferencias en el medio que se usa para transmitir, por ejemplo, la interferencia causada por otros sistemas, múltiples obstáculos (edificios, medio ambiente) en el camino, desvanecimiento por trayectos múltiples, etc… (Davies, 2011).

Algunos protocolos de transporte tales como el Protocolo de Control de Transmisión hacen un control de entrega mediante la recepción de un acuse de recibo de paquetes desde el receptor. Si los paquetes se pierden durante la transmisión, TCP volverá a enviar automáticamente los segmentos que no se reconoció a costa de disminuir el rendimiento general de la conexión.

Sin embargo, la sensibilidad de fiabilidad no es el mismo en todas las aplicaciones. Por ejemplo, es más importante que el correo electrónico, la transferencia de archivos y el acceso a Internet tengan transmisiones fiables en comparación de la telefonía o conferencias de audio (Bonaventure, 2011).

3.2.1.1 Tipos de pérdidas de paquetes

Muchas de las técnicas actuales para la recuperación de paquetes perdidos dependen en gran medida de cómo se dan estas pérdidas en la red (Chua & Pheanis, 2006). Es posible identificar dos perfiles diferentes de pérdida de paquetes: pérdida aleatoria y pérdida en ráfaga.

Una pérdida aleatoria significa una pérdida independiente, lo que implica que la pérdida de un paquete en particular es independiente de si recibieron o se perdieron los paquetes anteriores. Esta distribución de pérdidas es de la más estudiada en pruebas de calidad (Raake, 2006).

Por otro lado, la pérdida en ráfaga supone que una condición causante de la pérdida de un paquete persiste durante un cierto período de tiempo y por lo tanto nos hace perder dos o más paquetes consecutivos (Chua & Pheanis, 2006).

Tomando otra definición, en base al modelo Gilbert-Elliott, se dice que, una ráfaga o “burst” se presenta si en un período de tiempo limitado existe una tasa alta de pérdidas y/o descartes consecutivos, y un “gap” se presenta entre dos “bursts”. Los “gaps” son períodos de tiempo durante el cual existe nula pérdida de paquetes o bien, pueden ocurrir perdidas aleatorias de 1 paquete (T. Friedman, R. Caceres, & A. Clark, 2003). Por ejemplo, en la Imagen 3.1 se representa la ocurrencia de “gaps” y “bursts”, en esta imagen, un “0” representa un paquete perdido, un “1” representa un paquete recibido y la “X” representa un paquete descartado.

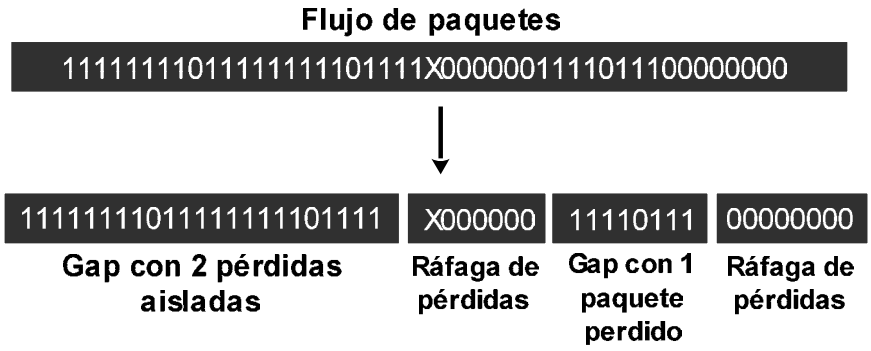


Imagen 3. 1 Diferencia entre ráfagas y gap

3.2.1.2 Técnicas de recuperación de pérdidas

La pérdida de paquetes suele ser consecuencia de un uso ineficiente del ancho de banda del enlace, como de una limitada velocidad de procesamiento del router. La solución más obvia a esto consistiría en mejorar la infraestructura de la red IP, los enlaces y los routers (QUERCIA, 2002). Pero esto no siempre resulta ser la mejor solución debido al elevado costo que podría significar el renovar los equipos, por tal motivo, se han desarrollado técnicas para atenuar los impactos negativos debido a la pérdida de paquetes. Dos de las más populares son: Forward error correction e Interleaving.

3.2.1.2.1 Forward error correction (FEC)

FEC es una técnica para la mitigación de los efectos no deseados de la pérdida de paquetes. Incluye información redundante en el flujo original y cuando se produce la pérdida de paquetes, la función de reparación de datos se utiliza para reconstruir aproximaciones o versiones exactas de algunos de los paquetes perdidos (QUERCIA, 2002). Existen muchas variantes diferentes de técnicas FEC, pero entre las más usadas están: Media-specific FEC y Media-independent FEC (Chua & Pheanis, 2006).

Media-specific FEC: este esquema envía un flujo de menor calidad como información redundante. En otras palabras, el i-ésimo paquete incluye tanto la i-ésima parte del flujo original y una copia de baja calidad del (i – 1)-ésimo parte (QUERCIA, 2002). Si se pierde un paquete, otro paquete que contiene la misma unidad será capaz de cubrir la pérdida.

La ventaja de este esquema es que utiliza medidas de cómputo de bajo costo y se puede codificar de forma compacta. Sin embargo, sólo puede cubrir períodos cortos de pérdida debido a la cruda naturaleza de las medidas.

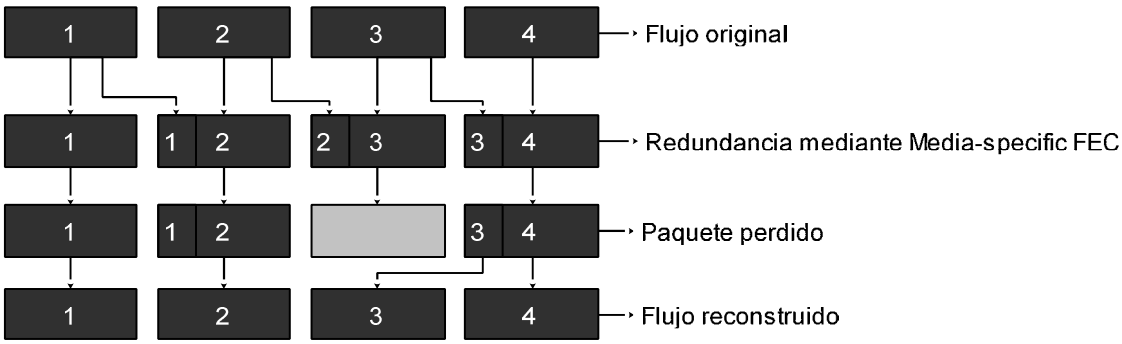


Imagen 3. 2 Media-specific FEC

Media-independent FEC: envía un bloque redundante después de cada *n* bloques. El bloque redundante se obtiene de los anteriores *n* bloques originales. Este uso de bloques, o códigos algebraicos para producir paquetes adicionales para la transmisión ayuda a la corrección de las pérdidas. Cada código toma una palabra de código de *k* paquetes de datos y genera *n-k* paquetes de comprobación adicional para la transmisión a través de la red (Perkins, Hodson, & Hardman, 1998). La Imagen 3.3 ilustra el principio.

Hay dos ventajas principales con este esquema: primero, la operación FEC no depende del contenido del paquete; y segundo, el esfuerzo computacional requerido para la FEC es relativamente pequeño (QUERCIA, 2002). Sin embargo, es difícil realizar una implementación de decodificador, además de agregar un retardo adicional y aumentar los requisitos de ancho de banda.

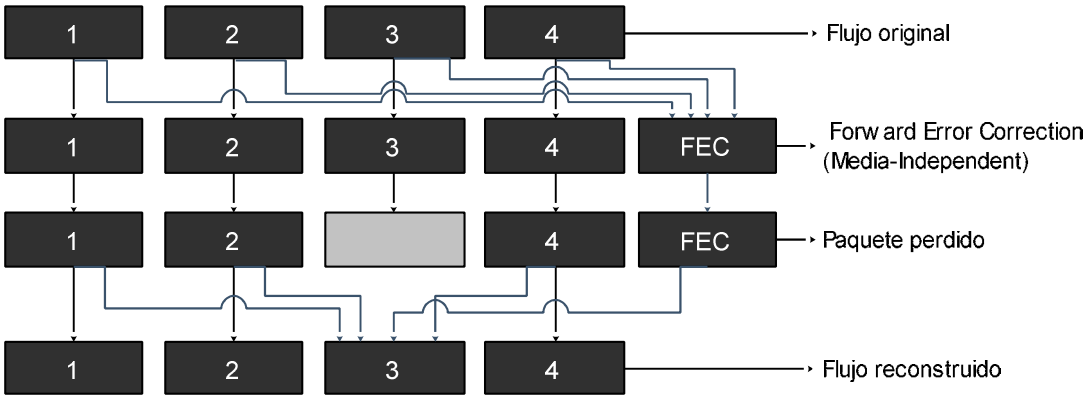


Imagen 3. 3 Media-Independent FEC

3.2.1.2.2 Interleaving

Este esquema no es precisamente una técnica de pérdida-recuperación, ya que este enfoque no es capaz de recuperar los datos perdidos. Sin embargo, este método intenta reducir la degradación de la calidad de transmisión perceptual mediante la distribución de la pérdida de datos en varios “gaps” pequeños en lugar de tener una gran ráfaga de datos perdidos (Chua & Pheanis, 2006).

Antes de la transmisión, a las tramas con contenido multimedia se les cambia la secuencia o entrelazan bajo un patrón específico y paquetizan en diferentes flujos. En el receptor se reconstruye de nuevo reacomodando las tramas transmitidas. Cuando se produce una pérdida de paquetes, el uso del flujo entrelazado ayuda a mitigar el efecto de las pérdidas consecutivas, de modo que se tendrán varios “gaps” pequeños en lugar de una gran secuencia de paquetes perdidos. Sin embargo, un inconveniente de esta técnica es el incremento en el retardo, debido a el tiempo necesario para reorganizar las tramas, tanto en el lado emisor y el lado receptor (Perkins, Hodson, & Hardman, 1998). La Imagen 3.4 ilustra un ejemplo de la técnica.

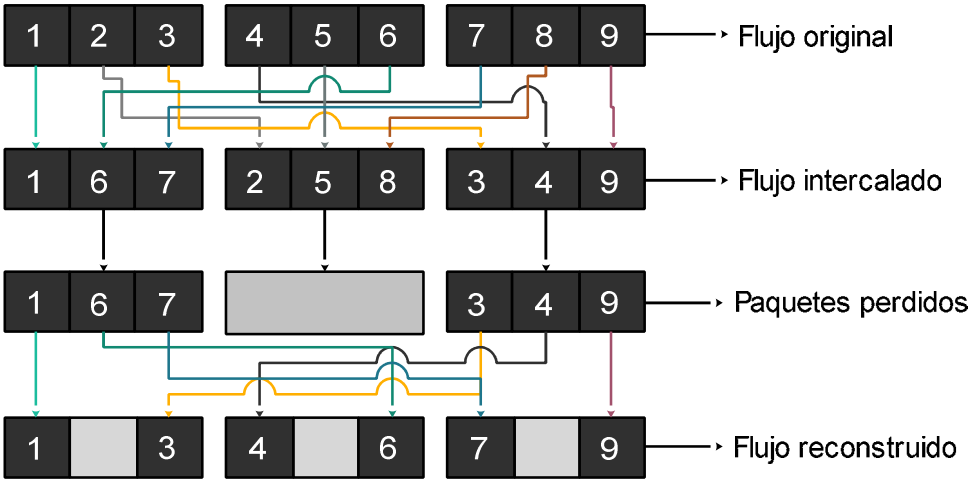


Imagen 3. 4 Interleaving

3.2.2 Retardo extremo a extremo

Este parámetro es característico en las comunicaciones, ya que los puntos finales son distantes y la información va a consumir algo de tiempo en alcanzar el otro extremo. A este parámetro también se le conoce como la latencia.

El tiempo de retardo puede aumentar si los paquetes se enfrentan a largas colas en la red (congestión), o cruza una ruta menos directa para evitar la congestión. El retardo se puede medir ya sea de un solo sentido (el tiempo total que tarda el paquete en ir del nodo fuente al destino), o de ida y vuelta (la latencia de un sentido

desde la fuente al destino, más la latencia de un sentido desde el destino a la fuente).

El retardo de ida y vuelta se utiliza con frecuencia, ya que se puede medir desde un único punto con el comando "ping". El retardo de ida y vuelta es una manera relativamente precisa de medir el retardo, ya que excluye la cantidad de tiempo que un sistema de destino gasta en procesar el paquete (Bellavista, 2009).

Las aplicaciones pueden tolerar el retardo en diferentes grados. En este caso, la telefonía, conferencias de audio, y videoconferencias, necesitan un retardo mínimo, mientras que el retardo en la transferencia de archivos o correo electrónico es menos importante o crítica (Bonaventure, 2011).

3.2.3 Jitter

El Jitter es la variación en el retardo de los paquetes que pertenecen al mismo flujo. Por ejemplo, si cuatro paquetes salen en los tiempos 0, 1, 2, 3 y llegan a los 20, 21, 22, 23, todos tienen el mismo retardo, 20 unidades de tiempo. Por otro lado, si los cuatro paquetes anteriores llegan a 21, 23, 21, y 27, tendrán diferentes retardos: 21,22, 19, y 24 (Forouzan, 2006).

El jitter es importante para muchas aplicaciones (por ejemplo, streaming de aplicaciones en tiempo real). Idealmente, los paquetes deben ser entregados de forma periódica, sin embargo, incluso si la fuente genera un flujo uniformemente espaciado, el jitter introducido por la red es inevitablemente, debido a la gestión de colas y la propagación de retardos variables (Mansour & Patt-Shamir, 2001).

Para hacer frente al jitter, es necesario hacer uso de buffers, es decir, recolectar los paquetes y mantenerlos el tiempo suficiente hasta que los paquetes más lentos lleguen, para luego reordenarlos en la secuencia correcta. Este procedimiento provoca un retardo adicional, pero es necesario en el caso de aplicaciones sensibles al jitter (Bellavista, 2009).

CAPÍTULO IV. STREAMING DE VIDEO

Internet está cambiando a un ritmo acelerado, y los usuarios que alguna vez se mostraron satisfechos con el texto e imágenes fijas en sus páginas web, cada vez desean más y mejor contenido, de acceso rápido y versátil.

El video ha sido un importante medio para las comunicaciones y el entretenimiento durante muchas décadas. Inicialmente era capturado y transmitido de forma analógica. Pero la constante evolución de las tecnológicas informáticas condujo a la digitalización de vídeo, y a iniciar toda una revolución en los métodos de compresión y transmisión de vídeo.

El streaming de video es una manera de transmitir vídeo a través de Internet. Aunque lejos de ser una solución perfecta, la tecnología de streaming de vídeo está convirtiéndola en una tecnología bastante popular debido al creciente ancho de banda a disposición de los usuarios. Con la transmisión de vídeo, es posible hacer anuncios, transmitir conferencias, ofrecer cursos, ver televisión por Internet o mostrar exactamente cómo funciona algo, dándoles una flexibilidad a los usuarios. Pueden ver lo que quieran ver y cuando lo quieran ver.

Este capítulo explorará los algoritmos y sistemas que permiten realizar streaming de vídeo en directo o pre-encoded sobre redes como Internet.

4.1. Arquitectura

Una arquitectura típica de streaming consta de 4 componentes básicos (Austerberry, 2013):

- Captura y codificación
- Almacenamiento en servidor
- Distribución y entrega
- Reproductor multimedia

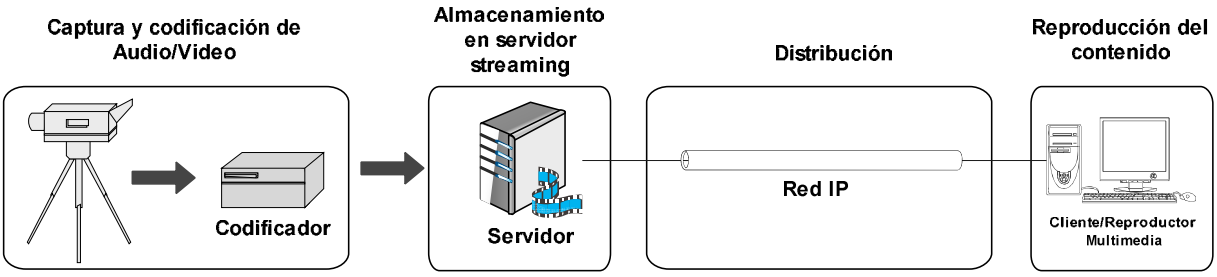


Imagen 4. 1 Arquitectura de video streaming

4.1.1. Captura y codificación

Se toman las señales de audio y de vídeo en formato sin procesar y se convierten a un archivo de transferencia. Hay una serie de etapas para esto:

- Capturar el archivo informático
- Comprimir los datos de medios
- Empaquetar los datos con pistas o índice

Una señal fuente de vídeo no puede ser manipulada por software hasta que esté en un formato compatible. Para ello, con una tarjeta de captura de vídeo instalada en el equipo de trabajo, se convierte tanto el video como el audio analógico o digital a un formato compatible, dando como resultado una reducción proporcional del tamaño de imagen. Esto genera una tasa de datos capaz de ser manejado fácilmente por el procesador.

Posteriormente se aplica un algoritmo de compresión que está incrustado en una aplicación de software llamado un codificador-decodificador, o CODEC. El codificador toma el archivo fuente de audio/vídeo y reduce la tasa de datos para que sea posible transmitirlo sobre un ancho de banda dado. El decodificador se encuentra en el reproductor multimedia, y rearma de nuevo el flujo de audio y vídeo para ser visualizado. El codificador también empaqueta los datos con un índice de metadatos que utiliza el servidor para controlar la entrega en tiempo real (Austerberry, 2013).

4.1.2. Almacenamiento en servidor

El archivo codificado se carga en un servidor que posteriormente se encargará de distribuirlo a una audiencia o usuario que lo solicite. El servidor suele ser una aplicación de software, como un servidor web, en vez de un servidor físico. Un servidor de streaming es algo más que un servidor de archivos, este es capaz de controlar en tiempo real la entrega del flujo de medios (Austerberry, 2013). Por lo tanto, los servidores de streaming utilizan protocolos distintos a los que usan los servidores web. Mientras que un servidor web utiliza el protocolo de transferencia de hipertexto, un servidor de streaming utiliza el protocolo de transmisión en tiempo real (RTSP) o protocolos propietarios como Microsoft Media Service (MMS). Estos protocolos están diseñados para demandas únicas de streaming, lo que requiere una conexión de dos vías, y la capacidad de controlar estrictamente el flujo (Mack & Rayburn, 2005). Más adelante se hablará de los protocolos utilizados en el streaming de medios.

4.1.3. Distribución y entrega

La idea principal de la distribución es simple. Mientras exista conectividad IP entre el servidor y el cliente, los paquetes solicitados llegarán al reproductor multimedia. Pero esto no es tan fácil, existen factores que pueden deteriorar la calidad final de un archivo multimedia; como que el vídeo esté distorsionado y el audio fuera de sincronización. El problema es que, originalmente, Internet no fue diseñado para soportar flujos continuos a través de conexiones continuas. En consecuencia, varios desarrollos diferentes están ayudando a mejorar la calidad de entrega. Como es el aumento gradual del ancho de banda que se ha venido presenciando en los últimos años. Otro punto que está ayudando a mejorar la calidad de transmisión son los protocolos de QoS, que ayudan a diferenciar el tráfico con prioridad (Austerberry, 2013).

La mayoría de las transmisiones se llevan a cabo a través de una entrega de unidifusión (unicast), donde a cada miembro de la audiencia se le envía un único flujo de medios. Pero esto resulta ser ineficiente en transmisiones por Internet, ya que a cada cliente se le debe enviar una copia única del flujo, a pesar de que existan más clientes viendo el mismo programa. Un método de entrega más eficaz es conocido como la multidifusión (multicast) que se puede utilizar para reducir el número de flujos de salida, véase Imagen 4.2 (Mack & Rayburn, 2005).

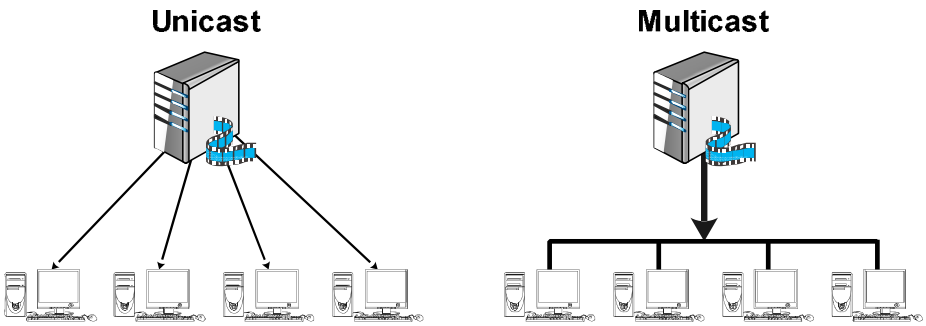


Imagen 4. 2 Unicast y multicast

4.1.4. Reproductor multimedia

El reproductor es el software del lado del cliente que se comunica con el servidor de medios, recibe el flujo entrante codificado, y lo decodifica para que sea visualizado. También proporciona al cliente el control sobre la reproducción del flujo. Diferentes reproductores de streaming pueden ofrecer distintas características, pero su funcionalidad básica es la misma (Mack & Rayburn, 2005).

4.2. Modos de acceso al medio

Existe una variada gama de diferentes aplicaciones de comunicación de vídeo y streaming, que tienen diferentes formas de operar y propiedades. Por ejemplo, aplicaciones de comunicación de vídeo pueden ser para una comunicación punto a punto, para multicast o broadcast, y el vídeo puede estar pre-codificado (almacenado) o puede estar codificado en tiempo real (por ejemplo, una videoconferencia interactiva). El acceso al medio puede ser mediante una descarga tradicional, bajo demanda (On-demand) o transmisión en vivo (Live streaming) (Apostolopoulos, Tan, & Wee, 2002).

4.2.1. Acceso por descarga tradicional

El usuario puede reproducir el archivo sólo después de que todo el archivo se ha descargado desde un servidor a su equipo. La transferencia completa, en este modo, a menudo puede sufrir largos tiempos de espera, que dependen del tamaño del archivo y el ancho de banda del canal de transporte. No impone requisitos especiales para el ancho de banda de red, pero a más ancho, más rápida la entrega (Austerberry, 2013).

4.2.2. Acceso bajo demanda (on-demand o descarga progresiva)

Es un paso intermedio entre la descarga tradicional y el live streaming. En lugar de esperar a que el archivo completo sea descargado en el disco local para luego reproducirlo, el archivo se puede ir visualizando mientras que el resto todavía se está descargando. La diferencia con el live streaming es que el servidor no está entregando el archivo en tiempo real, por lo que el reproductor puede quedarse sin material si la transferencia es más lenta que la velocidad de codificación, y todo el archivo multimedia se almacena localmente.

Los archivos multimedia previamente codificados se almacenan en un servidor y se entregan a uno o varios receptores cuando se solicite (bajo demanda). Actualmente varios sitios ofrecen streaming de audio y video almacenado, por ejemplo: Netflix, YouTube y Vimeo.

El vídeo pre-codificado tiene la ventaja de que no requiere una restricción en tiempo real de codificación. Esto puede permitir la codificación más eficiente. Pero por otra parte, su flexibilidad es limitada, por ejemplo, un vídeo pre-codificado no puede ser adaptado de manera significativa a los canales que soportan diferentes velocidades de bits o para clientes que admiten diferentes capacidades de visualización que la utilizada en la codificación original (Austerberry, 2013).

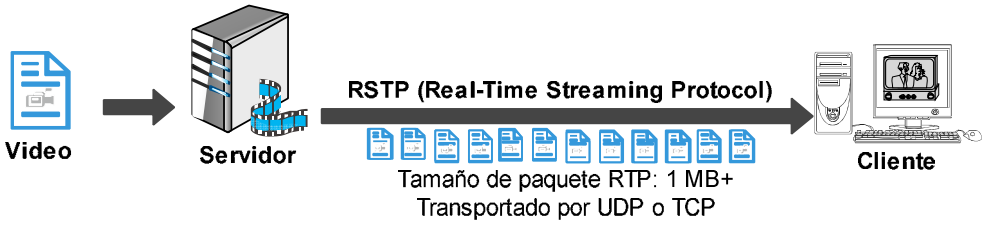


Imagen 4. 3 Descarga progresiva

4.2.3. Acceso en vivo (live streaming)

Con la transmisión en vivo, los medios a transmitir, son capturados, comprimidos y transmitidos todo en tiempo real. Este tipo requiere una cantidad significativa de recursos informáticos y a menudo soporte de hardware específico (Fecheyr-Lippens, 2010).

A medida que se reproduce el contenido de la transmisión en el lado del cliente, el reproductor multimedia ira descartándolo progresivamente después de que se haya reproducido; el contenido del streaming no existe tal cual como un archivo completo que se pueda guardar en el disco y reproducirlo posteriormente. Es como ver algo en la televisión. Se emite, lo ves y, al final de la transmisión, se va.

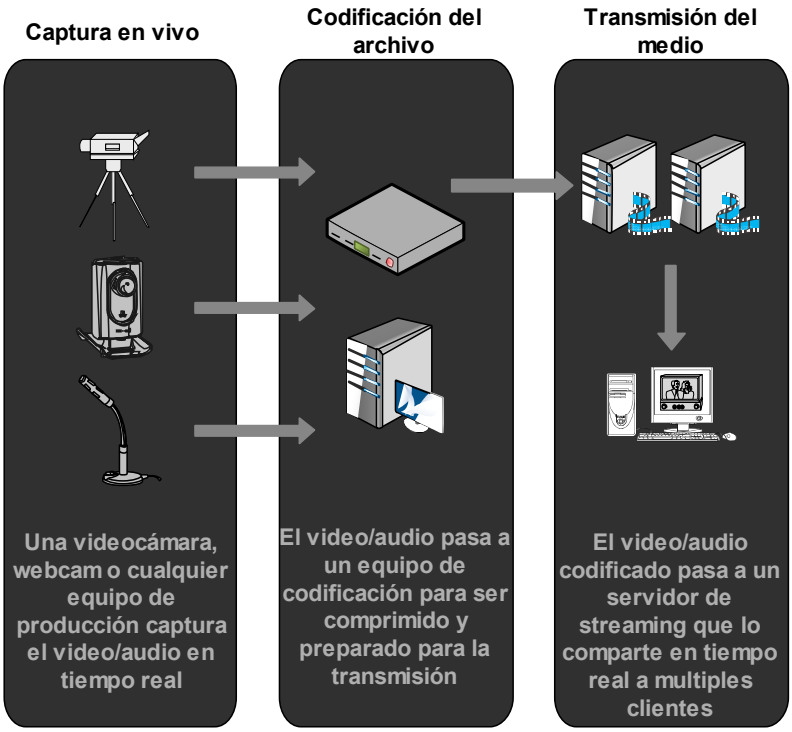


Imagen 4. 4 Transmisión en vivo

4.3. Protocolos de tiempo real (streaming)

Los protocolos están diseñados y estandarizados para la comunicación entre clientes y servidores de streaming. Los protocolos para la transmisión de medios ofrecen servicios tales como el direccionamiento de red, transporte y control de la sesión.

De acuerdo con sus funciones, los protocolos relacionados directamente con la transmisión de vídeo en Internet se pueden clasificar en las siguientes tres categorías (Wu D. , Hou, Zhu, Shanz, & Peha, 2001):

- Protocolo de capa de red
- Protocolo de transporte
- Protocolo de control de sesión

RTP es un protocolo de transporte para la transmisión de datos de medios mientras que RTCP es un protocolo para el control de la entrega de los paquetes RTP. UDP y TCP son protocolos de transporte de capa inferior para los paquetes de SIP/RTP/RTCP/RTSP, e IP proporciona una plataforma común para la entrega de paquetes UDP/TCP a través de Internet. La combinación de estos protocolos ofrece un servicio de streaming completo a través de Internet. En la Imagen 4.5 se observa como estos protocolos trabajan en conjunto para llevar a cabo el streaming de medios (Wu D. , Hou, Zhu, Shanz, & Peha, 2001).

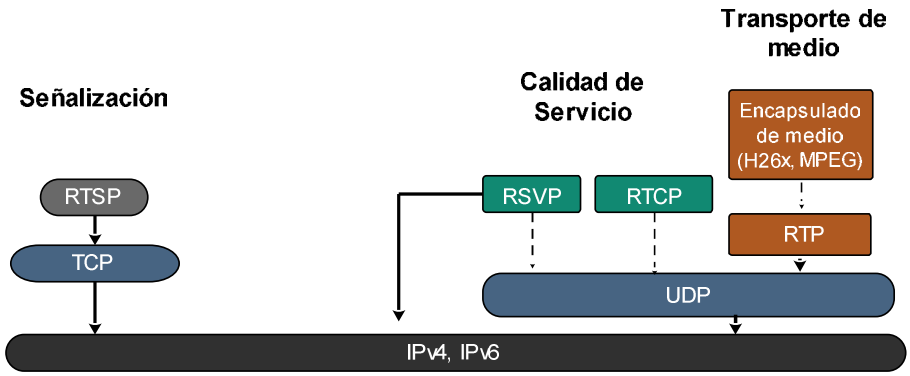


Imagen 4. 5 Protocolos para servicios de streaming

4.3.1. Protocolo de capa de red

Proporciona soporte de servicios básicos de red, como el direccionamiento de red. El protocolo de Internet (IP) es el protocolo de comunicación principal de la capa de red y es utilizado para la transmisión de vídeo por Internet. Pero IP tiene ciertos inconvenientes:

- Latencia de red variable
- Los paquetes pueden llegar en orden diferente de transmisión
- Los paquetes se pueden perder

Estos problemas se corrigen usando protocolos de capas superiores, como son TCP y UDP (Austerberry, 2013).

4.3.2. Protocolos de transporte

Brindan funciones de transporte en una red de extremo a extremo para aplicaciones de streaming. Los protocolos de transporte incluyen UDP, TCP, protocolo de transporte en tiempo real (RTP), y el protocolo de control en tiempo real (RTCP). UDP y TCP son protocolos de transporte de capa inferior mientras que RTP y RTCP son protocolos de transporte de capa superior, que se implementan junto con UDP/TCP (Wu D. , Hou, Zhu, Shanz, & Peha, 2001).

4.3.2.1. TCP

El protocolo de control de transmisión, es el más conocido en la capa de transporte utilizado junto con IP. Uno de sus puntos fuertes de TCP es su fiabilidad y el hecho de que está orientado a conexión, es decir, antes de transferir los datos, dos procesos de nivel de aplicación deben negociar formalmente una conexión TCP utilizando el proceso de establecimiento de conexión adecuado (Apostolopoulos, Tan, & Wee, 2002).

La protección contra errores hace que sea un protocolo excelente para la entrega de datos de propósito general, pero la forma en que esto se lleva a cabo resulta ser una desventaja para aplicaciones de streaming.

TCP aplica secuencias a los bytes de datos con un número de confirmación de reenvío que indica al destino el siguiente byte que espera recibir. Si los bytes no se recibieron dentro de un período de tiempo específico, se vuelven a retransmitir. Esta característica permite detectar la pérdida de paquetes y solicitar una retransmisión. Para evitar la transmisión de excesivos datos simultáneos, que podría causar problemas de congestión en la red, TCP proporciona control de flujo de datos (Apostolopoulos, Tan, & Wee, 2002).

4.3.2.2. UDP

Para la transmisión de vídeo y aplicaciones en tiempo real, la mayoría de los sistemas utilizan el Protocolo de Datagrama de Usuario (UDP) como protocolo de capa de transporte. UDP es adecuado para aplicaciones de flujo de vídeo y en tiempo real, ya que tiene menor demora en comparación con el Protocolo de Control de Transmisión. Debido a que UDP es un protocolo orientado a la no conexión, no es necesario establecer una conexión antes de enviar los paquetes a diferencia de TCP.

No cuenta con corrección de errores ni el control de flujo de TCP, por lo que esta tarea tiene que ser manejado por una aplicación en una capa superior. Sin embargo, siempre que la pérdida de paquetes no sea demasiado grande, UDP es el protocolo ideal para aplicaciones en tiempo real, ya que no todos los datos son fundamentales, debido a que los nuevos datos anulan los datos antiguos en aplicaciones de tiempo real (Austerberry, 2013).

4.3.2.3. RTP

El Protocolo de transporte en tiempo real, o RTP, es un método desarrollado específicamente para la transmisión de datos a través de redes IP tanto en aplicaciones de unicast o multicast. RTP normalmente se combina con RTSP para el transporte y puede trabajar sobre TCP o UDP.

RTP ofrece varios campos de datos que no están presentes en TCP, como una marca de tiempo (Timestamp) y un número de secuencia (Sequence number) para facilitar la sincronización de datos. Cuando RTP corre junto con UDP, utiliza las funcionalidades de multiplexación y de suma de comprobación. Permitiendo el control del servidor de comunicación para que se transmita el flujo de vídeo a la velocidad correcta (Schulzrinne, Casgner, Frederick, & Jacobson, 2003).

Los campos principales de RTP son:

Número de secuencia: es un valor que de 16 bits que se incrementa en uno por cada paquete. Es utilizado por el reproductor multimedia para detectar la pérdida de paquetes y luego secuenciar los paquetes en el orden correcto. El número inicial de una sesión de flujo se elige al azar (Schulzrinne, Casgner, Frederick, & Jacobson, 2003).

Marcas de tiempo o **Timestamps:** son una instancia de muestreo derivado de un reloj de referencia para permitir cálculos de sincronización y de jitter. Es monótona y lineal con el tiempo (Schulzrinne, Casgner, Frederick, & Jacobson, 2003).

Identificadores de fuente: CSRC es un identificador único para la sincronización del flujo RTP. Uno o más CSRC existen cuando el flujo RTP está llevando la información de múltiples fuentes de medios de comunicación. Este podría ser el caso de una mezcla de vídeo entre dos fuentes o de contenido incrustado (Schulzrinne, Casgner, Frederick, & Jacobson, 2003).

4.3.2.4. RTCP

El protocolo de control de transporte en tiempo real es un protocolo complementario de RTP, proporciona información de control a cada participante en una sesión de RTP y estos a su vez pueden utilizarlo para controlar la sesión.

RTCP se encarga de vigilar la calidad del servicio mediante mensajes que incluyen informes de recepción, incluyendo el número de paquetes perdidos y estadísticas de jitter. Esta información puede ser útil para las aplicaciones de capa superior, dándoles un criterio para que puedan modificar la transmisión (Schulzrinne, Casgner, Frederick, & Jacobson, 2003).

4.3.3. Protocolos de control de sesión

El Protocolo de transmisión en tiempo real es un protocolo de control de sesión para el streaming de medios a través de Internet. Una de las principales funciones de RTSP es soportar operaciones de control de tipo VCR como Pausar/Reanudar, Detener, Avanzar rápido y Retroceder rápido.

En RTSP, cada flujo de presentación y medios se identifica mediante un localizador uniforme de recursos RTSP (URL). La presentación general y las propiedades de los medios se definen en un archivo de descripción de presentación, que puede incluir el lenguaje de codificación, las direcciones URL RTSP, dirección de destino, el puerto y otros parámetros. El archivo de descripción de presentación se puede obtener por el cliente a través de HTTP, correo electrónico u otros medios.

Los flujos controlados por RTSP pueden utilizar RTP, pero la operación de RTSP no depende del mecanismo de transporte utilizado para transportar medios de comunicación continua. El protocolo es intencionalmente similar en sintaxis y funcionamiento de HTTP/1.1 de modo que los mecanismos de extensión HTTP pueden en la mayoría de los casos pueden ser añadidos a RTSP (Schulzrinne, Rao, & Lanphier, Real Time Streaming Protocol , 1998).

4.4. Compresión de video

El espacio y ancho de banda limitado puede ser un factor importante en la transmisión de video, sin embargo, existen algunos principios y prácticas de algoritmos dedicados a la compresión de vídeo que son especialmente relevantes para la comunicación y streaming.

Un vídeo sin procesar primero debe ser comprimido (codificado) antes de ser transmitido para lograr una mayor eficiencia. Los esquemas de compresión de vídeo se pueden clasificar en dos categorías: la codificación de vídeo escalable y no escalable.

La compresión del video es hecha por codificadores, que básicamente toman el audio entrante y/o vídeo, y lo codifican a una versión streaming con base a los parámetros que se le proporcionaron. Los codificadores utilizan matemáticas sofisticadas y modelos perceptuales para retener tanta fidelidad como sea posible. La fidelidad de un flujo codificado está directamente relacionada con la velocidad de bits: cuanto mayor es la velocidad de bits, mayor será la fidelidad (Mack & Rayburn, 2005).

La compresión se logra mediante la explotación de las similitudes o redundancias que existe en una típica señal de vídeo (Apostolopoulos, Tan, & Wee, 2002). Por ejemplo, los cuadros consecutivos en una secuencia de vídeo suelen mostrar redundancia temporal, ya que, normalmente contienen los mismos objetos. Otro objetivo de la compresión de vídeo es el de reducir la relevancia en la señal de vídeo, es decir, solo codificar funciones del vídeo que son perceptivamente importantes y no desperdiciar valiosos bits en información que no es perceptualmente importante o irrelevante. Identificar y reducir la redundancia en una señal de vídeo es relativamente sencillo, sin embargo, identificar lo que es perceptivamente relevante y lo que no es muy difícil y por lo tanto la irrelevancia es difícil de explotar (Apostolopoulos, Tan, & Wee, 2002).

4.4.1.1. Estándares de codificación

La codificación se lleva a cabo en varias etapas. La primera etapa consiste en capturar una señal de vídeo y convertirla a un formato de archivo que pueda ser procesado por el software de la computadora. La segunda etapa es la reducción de la tasa de datos mediante la ampliación y la compresión a una tasa de bits que puedan ser entregados a través de un acceso telefónico o circuitos de banda ancha. La tercera etapa es empaquetar el vídeo comprimido en un formato de paquetes que pueda ser transmitido a través de una red IP (Austerberry, 2013).

No existe una única forma de codificación; hay muchas maneras diferentes de obtener un vídeo en un formato de streaming. La opción que se elija dependerá del

hardware que ha seleccionado, el rendimiento requerido del material codificado, y la calidad de visión final que se requiera (Austerberry, 2013).

Los estándares de compresión proporcionan una serie de beneficios, sobre todo de los cuales se garantiza la interoperabilidad, o la comunicación entre los codificadores y decodificadores hechos por diferentes personas o empresas diferentes. De esta forma los estándares disminuyen el riesgo para el consumidor y el fabricante, y esto puede conducir a una aceptación más rápida y un uso generalizado (Apostolopoulos, Tan, & Wee, 2002).

Hay dos importantes familias de estándares de compresión de vídeo: la Unión Internacional de Telecomunicaciones (ITU-T) y la Organización Internacional de Normalización. En la Imagen 4.6 se observa los estándares importantes de cada familia.

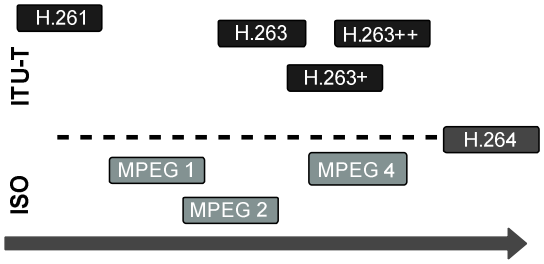


Imagen 4. 6 Estándares de compresión de video

4.4.1.2. H.261

El estándar H.261, también llamado Px64, es un algoritmo de compresión de movimiento desarrollado específicamente para videoconferencias a través de líneas telefónicas ISDN. Soporta dos formatos de imagen:

- Formato intermedio común (CIF), que tiene 352 x 288 píxeles para el canal de luminancia (Y) y 176 x 144 píxeles para cada uno de los dos canales de crominancia U y V. cuatro tasas temporales (30 15, 10 o 7,5 frames/seg). Imágenes CIF son usados para aplicaciones de videoconferencia.

- Un cuarto de formato intermedio común (QCIF) que tiene 176 x 144 píxeles para la Y y 88 x 72 píxeles para U y V respectivamente. Se utiliza normalmente para aplicaciones con tasas bajas de bits como teléfonos de video (UIT-T, 1993).

4.4.1.3. H.263

El estándar de vídeo H.263 fue diseñado para una codificación con baja velocidad de bits de secuencias de vídeo en videoconferencias. Al principio se pretendía ser utilizado en sistemas basados en H.323, pero también fue adoptado para videoconferencias basadas en Internet (Hanzo, Cherriman, & Streit, 2007).

Los algoritmos de codificación de H.263 son similares a los utilizados por su predecesor, es decir el CODEC H.261, aunque tanto su codificación, la eficiencia y la capacidad de recuperación de error se han mejorado. Algunos requisitos del estándar H.263 fueron:

- El uso de la tecnología disponible
- Interoperabilidad entre los otros estándares, como H.261
- Escalabilidad
- Parámetros de calidad de servicio, tales como la resolución, retardo, velocidad de fotogramas, etc.
- Mediciones subjetivas de calidad

4.4.1.4. H.264

H.264 es uno de los estándares de última generación que resultó de un proyecto conjunto entre el video de grupo de expertos de codificación de ITU-T y de la ISO/IEC Moving Picture Experts Group (MPEG) (Ponlatha & Sabeenian, 2013).

Entre sus objetivos está el proporcionar una buena calidad de vídeo a velocidades de bits sustancialmente más bajos que los estándares anteriores y con mayor

robustez a errores. Un objetivo adicional era proporcionar la suficiente flexibilidad para permitir que el estándar sea aplicado a una amplia variedad de aplicaciones: tanto para tasas de bits bajas y altas, para vídeo de baja y alta resolución, y con demandas de alta y baja latencia (Richardson I. E., 2003).

H.264 es el nombre utilizado por la ITU-T, mientras que la ISO/IEC utiliza el nombre MPEG-4 Parte 10/AVC, ya que se presenta como una nueva parte de su paquete de MPEG-4. La suite MPEG-4 incluye, por ejemplo, MPEG-4 Parte 2, que es un estándar que ha sido utilizado por los codificadores de vídeo basadas en IP y cámaras de red (Ponlatha & Sabeenian, 2013).

4.4.1.5. MPEG 1

El estándar MPEG-1 fue publicado 1992 y su objetivo era proporcionar una calidad VHS con un ancho de banda de 1,5 Mb/s, lo que permitió reproducir un vídeo en tiempo real desde un CD-ROM. La frecuencia de imagen en formato MPEG-1 está bloqueada a 25 fps (PAL) y 30 fps (NTSC), respectivamente. Además, MPEG-1 fue diseñado para permitir adelantar, retroceder, y sincronizar el audio y vídeo. Se logró un comportamiento estable, en los casos de pérdida de datos, así como bajos tiempos de cálculo para la codificación y decodificación (Richardson I. E., 2003).

El método de compresión de vídeo MPEG-1 combina elementos de JPEG, de H.261 y elementos nuevos significativos que permiten no sólo una compresión más alta, sino también puntos de entrada frecuentes en el flujo de vídeo. Dado que el vídeo es una secuencia de imágenes fijas, es posible lograr un poco de compresión utilizando técnicas similares a JPEG. Tales métodos de compresión comprimen o codifican cada imagen de vídeo de forma independiente.

Como en JPEG y H.261, el algoritmo de codificación de vídeo MPEG-1 emplea una de dos dimensiones DCT (transformada de coseno discreta) basado en bloques. Una imagen se divide primero en bloques de 8x8 elementos de imagen, y la DCT de dos dimensiones es entonces aplicada de forma independiente en cada bloque (Richardson I. E., 2003).

4.4.1.6. MPEG 2

El estándar MPEG-2 fue diseñado para proporcionar una compresión, codificación y transmisión de alta calidad. Específica los requisitos para la codificación de video, audio, sistemas de codificación para combinar audio y vídeo codificado con flujos de datos definidos por el usuario. Debido a que MPEG-2 fue diseñado como un estándar de transmisión, es compatible con una variedad de formatos de paquetes (incluyendo paquetes largos y de longitud variable de 1 kB hasta 64 kB), y proporciona capacidad de corrección de error que es adecuado para la transmisión por televisión por cable y los enlaces por satélite (Sikora, 1997).

MPEG-2 video fue dirigido a tasas de bits de más de 2 Mbits/s. En concreto, se diseñó originalmente para la codificación de video de alta calidad. Sin embargo, se amplió para incluir el vídeo de alta resolución, tales como la televisión de alta definición (HDTV).

El requisito principal del estándar fue lograr una alta compresión de vídeo entrelazado, manteniendo una calidad de vídeo alta. Con el fin de satisfacer todos estos requisitos MPEG-2 ha definido un gran número de funciones. Sin embargo, no todas las aplicaciones requieren todas las características de MPEG-2. Por lo tanto, para promover la interoperabilidad entre aplicaciones, MPEG-2 introdujo varios conjuntos de opciones algorítmicas (perfiles) y elección de los parámetros restringidos (niveles) para mejorar la interoperabilidad (Haskell & Puri, 2012).

En resumen, MPEG-2 está dirigido principalmente a aplicaciones con calidad de video con tasas de bits superior a 2 Mbit/s. Es adecuado para codificación de vídeo progresivo y entrelazado. Además, MPEG-2 también tiene compatibilidad y capacidad de ampliación con la norma MPEG-1. Su sintaxis es un súper conjunto de la sintaxis MPEG-1 y puede soportar una variedad de tipos y formatos para diversas aplicaciones. Al igual que en otras normas de codificación de vídeo, MPEG-2 sólo define la sintaxis y la semántica. No especifica cada opción de codificación (pre-procesamiento, estimación de movimiento, cuantificador, control de calidad

tasa, y otras opciones de codificación) y las opciones de decodificación (procesamiento posterior y la ocultación de errores) (Aramvith & Sun, 2005).

4.4.1.7. MPEG 4

MPEG-4 se basa y combina elementos de tres campos: la televisión digital, gráficos interactivos y la World Wide Web con el objetivo de no sólo proporcionar una compresión más alta, sino también un mayor nivel de interactividad con el contenido audiovisual, es decir, el acceso y la manipulación de los objetos en una escena visual (Aramvith & Sun, 2005). El estándar MPEG-4, o formalmente se inició en 1993, y el núcleo de la norma se completó en 1998. Desde entonces se han producido una serie de extensiones y adiciones a la norma.

En general, MPEG-4 aborda los siguientes requisitos principales:

- Acceso basado en el contenido
- Accesibilidad universal
- Compresión más alta

El estándar es capaz de soportar todas las funcionalidades de MPEG-1 y MPEG-2. MPEG-4 mejora varios aspectos de MPEG-2, tanto en términos de la eficiencia de compresión alcanzable, así como en términos de flexibilidad lo que facilita su empleo en una amplia gama de aplicaciones (Hanzo, Cherriman, & Streit, 2007).

4.5. Parámetros QoS para video streaming

En la sección 3.2 se habló sobre los parámetros de QoS que intervienen en la transmisión de datos, ahora veremos cómo estos parámetros afectan la calidad de una transmisión de video streaming.

Retardo: una transmisión de video requiere de un retardo de extremo a extremo limitado, para que los paquetes puedan llegar al receptor a tiempo, ser decodificados y reproducidos. Si un paquete no llega a tiempo, la reproducción de video se detendrá. Un paquete de video que llega más allá del límite de retardo obligado, resulta inútil y puede ser considerado como perdido (Wu D. , Hou, Zhu, Shanz, & Peha, 2001).

Pérdida: La pérdida de paquetes es inevitable en Internet y puede dañar la calidad de imagen del video. Por lo tanto, es conveniente que un flujo de vídeo sea robusto para la pérdida de paquetes. La descripción múltiple de codificación es una técnica de compresión para hacer frente a la pérdida de paquetes (Wu D. , Hou, Zhu, Shanz, & Peha, 2001).

Jitter: El retardo de extremo a extremo que un paquete experimenta puede variar de un paquete a otro. Esta variación se conoce como fluctuación de retardo o jitter. El jitter es un problema debido a que el receptor debe recibir/decodificar/visualizar cuadros (frames) a una velocidad constante, y cualquier cuadro atrasado por culpa del jitter puede producir problemas en la reconstrucción de vídeo, por ejemplo, espasmos en el vídeo. Este problema normalmente se minimiza al incluir un buffer temporal de reproducción en el receptor. Para que el buffer de reproducción pueda compensar la fluctuación de retardo, también introduce un retardo adicional (Apostolopoulos, Tan, & Wee, 2002).

Las aplicaciones de video streaming por lo general tienen requerimientos de calidad de servicio bastante tolerantes, ya que no son tan sensibles al retardo (el video puede tardar varios segundos en comenzar a reproducirse) y son en gran medida insensibles al jitter (debido al almacenamiento en búfer de las aplicaciones). Sin embargo, el streaming de vídeo puede contener contenido valioso, como aplicaciones de e-learning o reuniones de empresa de multidifusión, y por lo tanto pueden requerir garantías de servicio.

Se recomienda las siguientes pautas de QoS para un tráfico de streaming de vídeo (Szigeti, Hattingh, & Barton, 2013):

- Las pérdidas de paquetes no deben superar el 5%
- La latencia debe ser de no más de 4-5 segundos (dependiendo de la capacidad de buffer de la aplicación de vídeo)
- No existen requisitos significativos de jitter, pero se recomienda que este entre el rango de 0 - 30 ms
- El ancho de banda garantizado depende del formato de codificación y la velocidad del flujo de vídeo

CAPÍTULO V. MÉTODOLOGIA DE MEDICIÓN

Con el fin de proporcionar una alta calidad de experiencia (QoE) para los usuarios finales, la información del rendimiento de la red, debe ser monitoreado y bien analizado a través de observaciones. Los resultados de la observación de una métrica específica deben compararse con referencia a los resultados estables que están previamente regulados. De este modo, una observación se convierte en una medición. La precisión de las mediciones tiene un gran efecto sobre la calidad de transferencia de datos, por lo que estos resultados ayudan a producir e interpretar descripciones cuantitativas de atributos útiles para la toma de decisiones críticas.

5.1. Tipos de medición

Las mediciones se llevan a cabo utilizando diversos enfoques, incluso dos de ellos se basan en los mismos parámetros y datos durante el proceso. Los enfoques más comunes son la medición activa y la medición pasiva. Para este caso de estudio, se ha hecho uso de un enfoque pasivo.

5.1.1. Medición activa

Esta se realiza mediante la generación e inyección de tráfico de prueba en la red, para medir, por ejemplo, el tiempo que tarda un paquete en llegar al otro extremo de la red, la respuesta de tiempo de una aplicación o la disponibilidad de una ruta de la red (Mohan, Reddy, & Kalpana, 2011).

A diferencia de las mediciones pasivas, las mediciones activas generan tráfico de red adicional por lo que posiblemente pueden perturbar el flujo normal del tráfico.

Tradicionalmente, las herramientas de medición activan más usadas son Ping y Traceroute para determinar retardos de ida y vuelta y topologías de red que utilizan paquetes ICMP (Calyam, Krymskiy, Sridharan, & Schopis).

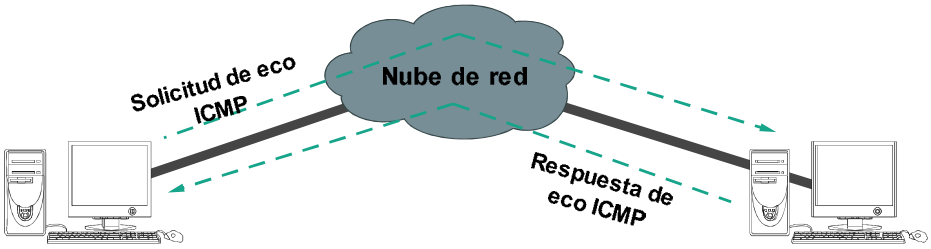


Imagen 5. 1 Medición activa

5.1.2. Medición pasiva

En las mediciones pasivas, los datos son recopilados mediante el monitoreo pasivo del tráfico de red, es decir, no se inyectan paquetes de prueba en la red (Mohan, Reddy, & Kalpana, 2011).

Requiere de la captura de paquetes y sus correspondientes marcas de tiempo transmitidos por las aplicaciones que se ejecutan en los dispositivos conectados a la red (por ejemplo, conmutadores y routers) a través de varias rutas de red (Calyam, Krymskiy, Sridharan, & Schopis).

Ethereal (hoy en día llamado Wireshark) y tcpdump son algunas de las herramientas de medición pasiva que más se utilizan.

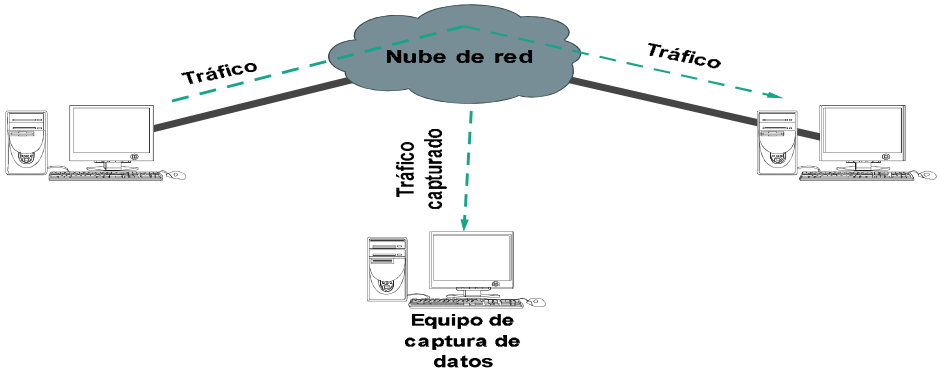


Imagen 5. 2 Medición pasiva

5.2. Escenario de medición

El escenario de medición utilizado en el presente trabajo está constituido por una arquitectura cliente-servidor con capacidad de transmisión de video streaming. Esta arquitectura está formada por dos redes WLAN, geográficamente separadas a una distancia aproximada de 4 km. e interconectadas por un enlace punto a punto mediante dos antenas con tecnología MIMO. La red WLAN-A está ubicada en la Universidad de Quintana Roo, División de Ciencias de la Salud (DCS) y la red WLAN-B en la Universidad de Quintana Roo, en el Centro de Tecnologías de la Información y la Comunicación (CTIC), véase Imagen 5.3.

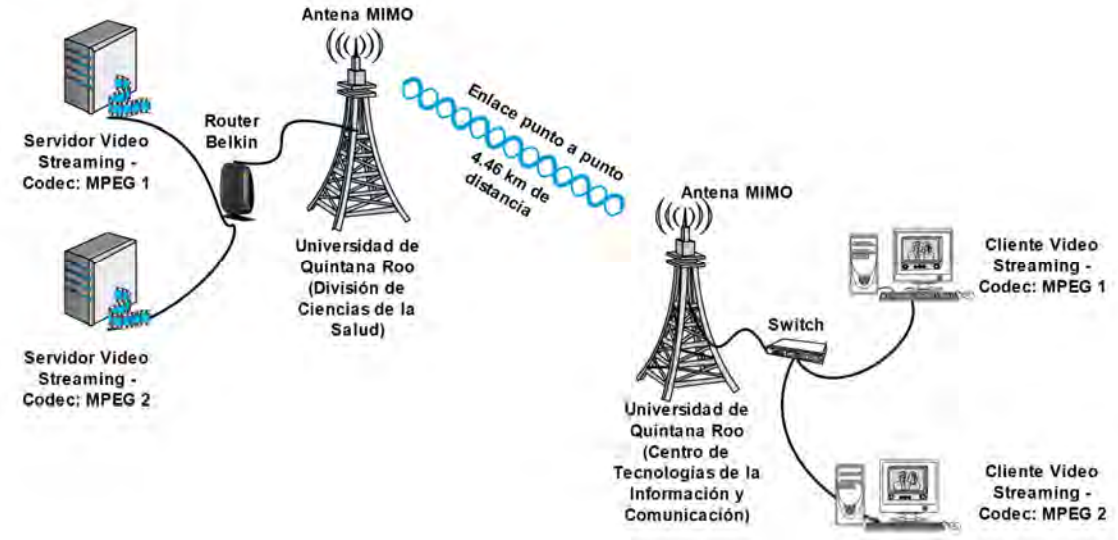


Imagen 5. 3 Escenario de medición

WLAN-A. Está ubicada en el Área de Computo de la DCS, cuenta con 2 computadoras de escritorio. Estas están conectadas por medio de cable Ethernet a un router (Belkin N600) que a su vez está conectado por un cable Ethernet a una antena MIMO (Antena MIMO DCS), la cual se encuentra montada en la torre de comunicaciones de la División de Ciencias de la Salud UQROO.

WLAN-B. Está ubicada en el Centro de Tecnologías de la Información y la Comunicación, cuenta con 2 computadoras de escritorio que se encuentran conectadas por medio de cable Ethernet a un Switch que a su vez está conectado a una antena MIMO (Antena MIMO DCI), la cual se encuentra montada en la torre de comunicaciones del Centro de Tecnologías de la Información y la Comunicación.

Las computadoras de escritorio de cada LAN tienen una configuración específica para la transmisión de video streaming.

En la **WLAN-A** se ha configurado cada PC como un servidor de video streaming con la aplicación **VLC**. La **PC1**, se le designó como servidor de video streaming que transmitirá el contenido multimedia utilizando el CODEC **MPEG1**, mientras que la **PC2** transmitirá el mismo contenido utilizando el CODEC **MPEG2**.

En la **WLAN-B**, las dos PC's se configuraron como clientes de video streaming, utilizando también la aplicación **VLC**. La **PC1** recibirá la transmisión de video que utiliza el CODEC **MPEG1**, mientras que la **PC2** recibirá la transmisión de video bajo el CODEC **MPEG2**.

Se realizó una medición pasiva en la WLAN-B utilizando un analizador de paquetes y protocolos de red, específicamente la aplicación **Wireshark**, posteriormente se hizo un análisis de los paquetes capturados mediante la aplicación R y la ejecución de scripts desarrollados para el tratamiento de estos datos en específico.

5.2.1. Características de los equipos

El escenario global, cuenta con los siguientes equipos:

- 4 Computadoras HP Compaq 6000 Pro con las mismas características. Véase en Tabla 5.1.
- 1 Router inalámbrico. Véase Tabla 5.2.
- 1 Switch. Véase Tabla 5.3.
- 2 Antenas MIMO Ubiquiti NanoStation M5. Véase Tabla 5.4.

Computadora HP Compaq 6000 Pro SFF PC
Windows XP Professional
Procesador: Pentium(R) Dual-Core CPU E5700 @ 3.00GHz
Memoria (RAM): 4.00 GB
Disco duro: 512 Gb
Gráficos: Intel(R) Q45/Q43 Express Chipset (Microsoft Corporation - WDDM 1.1)

Tabla 5. 1 Información de computadoras

Router Inalámbrico Belkin N600 DB
Transmisión de doble banda (Dual Band)
Hasta 600Mbps (hasta 300 Mbps (2,4GHz) + 300 Mbps (5 GHz))
Tecnología MultiBeam: Cobertura total para varios dispositivos
1 puerto USB: Para la impresión o el almacenamiento inalámbrico
4 puertos LAN Ethernet: Para conexiones de red cableadas rápidas y fiables
Diseño de la antena: Interna e integrada
Compliant Standard(s): IEEE 802.11b, IEEE 802.11g y IEEE 802.11n
Ancho de banda: radio dual concurrente 2,4 GHz y 5 GHz ISM
Conmutador integrado: Conmutador de 4 puertos

Tipo de conector(s): USB de 4 patillas tipo A (1)
Tipo de Interfaz: RJ-45 (LAN)
Interfaz: Ethernet 10Base-T/100Base-TX (4)
Nº máx. de conexiones WLAN: 16
Navegadores compatibles para gestión remota: Firefox® y Safari®
Protocolo de conmutación: Ethernet
Protocolo de enrutamiento: Enrutamiento de IP estática
Protocolo de gestión remota: HTTP y HTTPS
Protocolo(s) de enlace de datos: Ethernet, Fast Ethernet, IEEE 802.11a, IEEE 802.11b, IEEE 802.11g y IEEE 802.11n
Protocolos de ISP compatibles: Estática (IPoA), PPPoA, PPPoE
Velocidad de transferencia de datos: 300 Mbps
Método(s) de autenticación: Identificación de conjunto de servicios de radio (SSID)
Seguridad por pulsador WPS: Si
Tipo(s) de encriptación compatible(s): WPA de 256 bits, WEP de 64 bits y WPA2
Tipo(s) de firewall compatible(s): NAT y SPI

Tabla 5. 2 Router inalámbrico Belkin N600 DB

Switch GREENnet Gigabit de 24 puertos
24 puertos Gigabit
Malla de conmutación de 48 Gbps
Compatible con Jumbo Frame
Ethernet: 10 Mbps (half dúplex), 20 Mbps (full dúplex)
Fast Ethernet: 100 Mbps (half dúplex), 200 Mbps (full dúplex)
Gigabit: 2000 Mbps (full dúplex
Fuente de alimentación interna: 100 - 240 V AC, 50/60 Hz, 0.2 A

Certificación: CE, FCC

Tabla 5. 3 Información de switch

Antena MIMO Ubiquiti NanoStation M5
5GHz Hi Power 2x2 MIMO AirMax TDMA Station
Más de 150Mbps de velocidad
Alcance de 15Km
2x2 MIMO
Antena de 16dBi con 500mW de potencia
Trabaja con 5GHz de Frecuencia
Procesador: Atheros MIPS 24KC, 400MHz
Memoria: 32MB SDRAM, 8MB Flash
Interfaz de Red: 2 x 10/100 BASE-TX (Cat. 5, RJ-45) Interface Ethernet
Conformidad ROHS: SI
Peso: 0.4kg
Características de la caja: Exterior, Plástico UV estabilizado
Máximo consumo de energía: 8 Watios
Alimentación: UBIQUITI POE-2412W Fuente de alimentación PoE incluido
Rango de Frecuencia: 4.9-5.9 GHz
Ganancia: 14.6-16.1dBi
Polarización: Lineal Dual
Aislamiento de Polaridad: 22dB Mínimo
Frecuencia: 5470MHz-5825MHz

Tabla 5. 4 Antena MIMO Ubiquiti NanoStation M5

5.2.2. Direccionamiento IP

Se utilizó un direccionamiento IPv4, a continuación, se brinda las tablas con las correspondientes direcciones IP asignadas a cada equipo.

5.2.2.1. LAN-A (DCS)

Dirección IP	**192.168.1.33**
Máscara de subred	255.255.255.0
Servidor MPEG-1	

Tabla 5. 5 Dirección IP PC1_MPEG1_Server

Dirección IP	**192.168.1.34**
Máscara de subred	255.255.255.0
Servidor MPEG-2	

Tabla 5. 6 Dirección IP PC2_MPEG2_Server

Ajustes LAN	
Dirección IP	192.168.1.13
Máscara de subred	255.255.255.0
Servidor DHCP	Activado
Ajustes de Internet	
Tipo de conexión	Estática
IP de WAN	192.168.6.83
Máscara de subred	255.255.255.0
Puerta de enlace predeterminada	192.168.6.254
Dirección DNS	8.8.8.8

Tabla 5. 7 Router Belkin N600 – DCS

Wireless Mode	Cliente WDS
Dirección IP	192.168.1.14
Máscara de subred	255.255.255.0
Puerta de enlace predeterminada	192.168.6.83
Servidor DNS	8.8.8.8
SSID	ubntuqroo

Tabla 5. 8 Antena MIMO – DCS

5.2.2.2. LAN-B (CTIC)

Dirección IP	**192.168.1.140**
Máscara de subred	255.255.255.0
Cliente MPEG-1	

Tabla 5. 9 Dirección IP PC1_MPEG1_Cliente

Dirección IP	**192.168.1.141**
Máscara de subred	255.255.255.0
Cliente MPEG-2	

Tabla 5. 10 Dirección IP PC2_MPEG2_Cliente

Wireless Mode	Estación WDS
Dirección IP	192.168.1.15
Máscara de subred	255.255.255.0
Puerta de enlace predeterminada	192.168.6.83
Servidor DNS	8.8.8.8
SSID	ubntuqroo

Tabla 5. 11 Antena MIMO - CTIC

5.2.3. Software utilizado

En la tabla 5.12 se presenta las características técnicas del software usado en el proceso de transmisión, captura y análisis del flujo multimedia, posteriormente se hablará brevemente de las funciones que realiza cada uno.

Nombre	Licencia	Versión	Tipo
VLC	Gratuita	2.2.1	Reproductor multimedia, codificador y transmisor.
Wireshark	Gratuita	1.12.5	Captura y analiza paquetes y protocolos de red
R	Gratuita	3.2.1	Manipulación y gráficos de datos estadísticos.

Tabla 5. 12 Software usado

5.2.3.1. VLC

VLC media player es un reproductor multimedia y framework multimedia libre y de código abierto desarrollado por el proyecto VideoLAN. Es un programa multiplataforma con versiones disponibles para muchos sistemas operativos, es capaz de reproducir casi cualquier formato de video sin necesidad de instalar CODECs externos y puede reproducir videos en formatos DVD, Bluray, a resoluciones normales, en alta definición o incluso en ultra alta definición o 4K.

VLC es un reproductor de audio y video capaz de reproducir muchos CODECs y formatos de audio y video, además de capacidad de streaming. Es software libre, distribuido bajo la licencia GPL (VideoLAN Organization).

5.2.3.2. Wireshark

Wireshark es un analizador de protocolos basado en las librerías pcap utilizado comúnmente como herramienta de diagnóstico de redes y de desarrollo de aplicaciones de red.

Entre sus cualidades nos encontramos con una enorme versatilidad que le lleva a soportar más de 480 protocolos distintos, además de la posibilidad de trabajar tanto con datos capturados desde una red durante una sesión como con paquetes previamente capturados que hayan sido almacenados en el disco duro.

Soporta el formato estándar de archivos tcpdump, es capaz de reconstruir sesiones TCP, y está apoyado en una completa interfaz gráfica que facilita enormemente su uso (Wireshark Foundation).

5.2.3.3. R

R es un lenguaje y entorno para computación de estadísticos y gráficas. Es un proyecto GNU, que fue desarrollado en los Laboratorios Bell (antes de AT&T, ahora Lucent Technologies) por John Chambers y sus colegas.

R ofrece una amplia variedad de técnicas estadísticas (modelado lineal y no lineal, pruebas estadísticas clásicas, análisis de series temporales, clasificación, agrupación,) y gráficas, y es altamente expandible.

R está disponible como software libre bajo los términos de la Licencia Pública General de GNU de la Free Software Foundation en forma de código fuente. Se compila y se ejecuta en una amplia variedad de plataformas UNIX y sistemas similares (incluyendo FreeBSD y Linux), Windows y MacOS (The R Foundation).

5.3. Mediciones

Durante 4 días se transmitió un flujo constante de video utilizando tanto el CODEC MPEG-1, como el CODEC MPEG-2, los dos de forma simultánea. Las mediciones de cada día tuvieron una duración de 12 horas cada uno, iniciando a las 9:00 horas y finalizando a las 21:00 horas. A cada día se le dedicó un solo ancho de banda de los soportados por las antenas MIMO (40 MHz, 20 MHz, 10 MHz y 5 MHz).

Día	Ancho de banda	CODEC	Horas
1	40 MHz.	MPEG-1 – MPEG-2	12 Horas
2	20 MHz.	MPEG-1 – MPEG-2	12 Horas
3	10 MHz.	MPEG-1 – MPEG-2	12 Horas
4	5 MHz.	MPEG-1 – MPEG-2	12 Horas

Tabla 5. 13 Itinerario de medición

Los 4 días de transmisión fueron capturados utilizando Wireshark en el lado del cliente. Definiendo que un set de medición, es el total de paquetes capturados en un intervalo de tiempo dado, y, que las 12 horas de medición se capturaron en intervalos de 60 minutos, tenemos que, en 1 día de medición, considerando que se utilizaron los dos CODEC de forma simultánea y que una transmisión se divide tanto en un flujo de video y uno de audio, se obtuvieron un total de 48 sets de paquetes capturados (24 sets por cada CODEC). Para una mejor visualización observe la tabla 5.14

Sets de paquetes capturados					
CODEC	**Día 1 40 MHz**	**Día 2 20 MHz**	**Día 3 10 MHz**	**Día 4 5 MHz**	**Total de sets por CODEC**
MPEG-1 Audio	12	12	12	12	48
MPEG-1 Video	12	12	12	12	48
MPEG-2 Audio	12	12	12	12	48
MPEG-2 Video	12	12	12	12	48
Total de sets por día	48	48	48	48	

Tabla 5. 14 Archivos capturados

5.3. Extracción de datos

Sumando los totales de la tabla 5.14, tenemos un total de 96 sets de mediciones divididos en 48 archivos. pcap capturados por Wireshark (1 archivo. pcap almacena tanto un flujo de video, como uno de audio). La apertura de esta cantidad de archivos. pcap, sumado a la selección y extracción de ciertos parámetros de interés, se torna tedioso, lento y aumenta la posibilidad de cometer un error humano al momento de guardar los datos.

Para evitar la serie de complicaciones mencionadas anteriormente, se desarrolló un sencillo script junto con TShark, una versión de Wireshark orientada a línea de comandos, para extraer y exportar los parámetros a un archivo .csv.

El comando principal del script se aprecia a continuación:

```
tshark -Y "rtp.p_type == 32" -T fields -n -r "C:\Carpeta_de_Archivos\archivo1.pcap" -E header=y -E separator=, -e
frame.len -e udp.length -e udp.srcport -e rtp.seq -e frame.time_epoch -e rtp.timestamp -e rtp.seq  >> "C:\
Carpeta_destino\Parametros_v_1.csv"
```

```
tshark -Y "rtp.p_type == 14" -T fields -n -r "C:\Carpeta_de_Archivos\archivo1.pcap" -E header=y -E separator=, -e
frame.len -e udp.length -e udp.srcport -e rtp.seq -e frame.time_epoch -e rtp.timestamp -e rtp.seq  >> "C:\
Carpeta_destino\Parametros_A_1.csv"
```

En donde:

rtp.p_type hace referencia al tipo de flujo deseado, 32 para video y 14 para audio.

-e indica cada uno de los parámetros que se desean extraer del flujo elegido.

Ejecutado el script, tenemos ahora un total de 96 archivos .csv conteniendo los parámetros necesarios para el análisis de los flujos de streaming.

Para ver el script completo, diríjase a la sección de ANEXO A.1.

5.4. Procesamiento de datos

Por cada set de medición se tiene un promedio de 500 mil paquetes capturados, tomando en cuenta esto, puede resultar cansado analizar una cantidad masiva de datos por separado. Por tal motivo, se utilizó R, un lenguaje y entorno para computación de estadísticos y gráficas, para la creación, manipulación y análisis de una base de datos cuya información almacenada son los parámetros extraídos de los archivos .pcap. Utilizando R como lenguaje de programación se desarrollaron una serie de scripts para el cálculo de métricas nuevas a partir de los parámetros extraídos de Wireshark. Estás nuevos datos son: jitter, paquetes perdidos, “bursts” y “gaps”.

5.4.1. Cálculo de jitter

El cálculo del parámetro jitter, se hizo con base al **RFC3550 (RTP),** para ello se utilizaron las métricas extraídas de wireshark (rtp.timestamp, frame.time_epoch), el RTP timestamp clock rate del CODEC y las siguientes formulas:

$$D_{(i,j)} = (R_j - R_i) - (S_j - S_i) = (R_j - S_j) - (R_i - S_i)$$

Fórmula 5.1

En donde:

- $D(i,j)$ es la diferencia de tiempo entre la llegada del paquete i y el paquete j
- R es el tiempo de llegada del paquete en segundos
- S es la estampa de tiempo (rtp.timestamp) del paquete basándose en la frecuencia de muestreo, también llamado tiempo nominal

$$J(i) = J_{(i-1)} + (|D_{(i-1,i)}| - J_{(i-1)})/16$$

Fórmula 5.2

En donde:

- $J(i)$ es el jitter del paquete
- $J(i-1)$ es jitter del paquete anterior
- $|D(i-1,i)|$ es el valor absoluto de la diferencia de tiempo calculado con la fórmula 5.1

Este cálculo se automatizó con un script escrito en R, diríjase a la sección de ANEXO A.2 para más información.

5.4.2. Cálculo de paquetes perdidos

Para poder detectar tanto los paquetes RTP esperados y los perdidos, es necesario analizar el estado de los números de secuencia (rtp.seq)

Los números de secuencia se incrementan de uno en uno por cada paquete RTP enviado, y son utilizados por el receptor para detectar las pérdidas y restablecer la secuencia de paquetes. El valor inicial del número de secuencia es al azar (impredecible), pudiendo tomar cualquier valor inicial en el rango de 0 a 65535.

El proceso llevado a cabo para el cálculo de pérdidas en este trabajo consiste en verificar la diferencia entre el número de secuencia del paquete recibido actual y el número de secuencia del paquete anterior, si la diferencia es superior a 1, se entiende que ha habido una pérdida de paquete o paquetes.

En caso de que ocurra una pérdida, se marcan los paquetes faltantes con un 0, para posteriormente contabilizarlos. En la Imagen 5.4 se puede observar el diagrama de flujo que describe básicamente todo el proceso y en el ANEXO A.3 encontrará el script escrito en R.

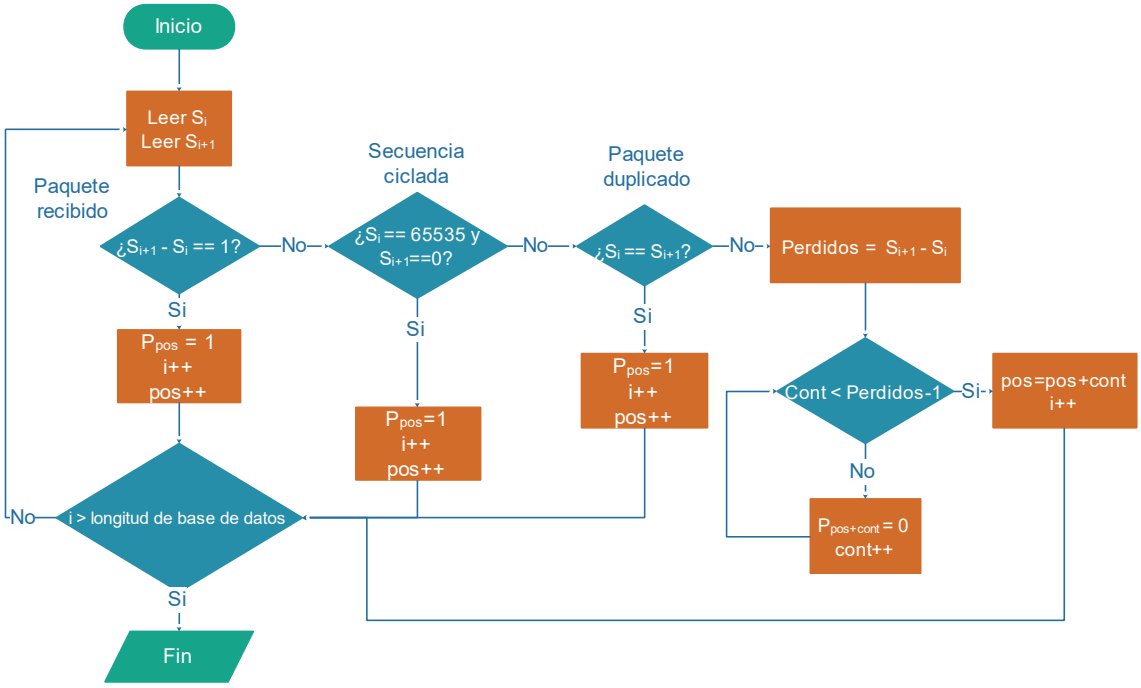


Imagen 5. 4 Cálculo de paquetes perdidos

5.4.3. Cálculo de “bursts” y “gaps”

Hay dos tipos de pérdida de paquetes: "aleatoria" y "ráfagas". La pérdida de un paquete de manera aislada no suele impactar considerablemente en la calidad de una transmisión, sin embargo, las cosas cambian cuando se comienzan a perder dos o más paquetes consecutivos, esto puede ser indicios de algún problema en la red, que al final terminarán por degradar la calidad de la transmisión. Siguiendo esta idea, se entiende por ráfaga o “burst”, al evento en donde dos o más paquetes se pierden de manera consecutiva. Con base a la definición anterior, una ráfaga se caracteriza por tener un tamaño y una frecuencia. El tamaño se define como la cantidad de paquetes perdidos de manera consecutiva, mientras que la frecuencia describe la cantidad de veces en que se presenta la esta misma ráfaga.

Un “gap” para propósitos de este trabajo será definido como la cantidad de paquetes recibidos consecutivamente posteriores a un evento de ráfaga y antes de la aparición de otra ráfaga.

Recordemos que para este trabajo se utiliza el valor 0 para referirse a un paquete perdido y el valor 1 para un paquete recibido. En la Imagen 5.5 puede observar el procedimiento básico seguido para el cálculo de ráfagas utilizando la información previamente recabada en el proceso de cálculo de pérdidas. Para “gaps” se siguió el mismo proceso, solo que contabilizando los 1 Para ver el script completo diríjase al ANEXO A.4

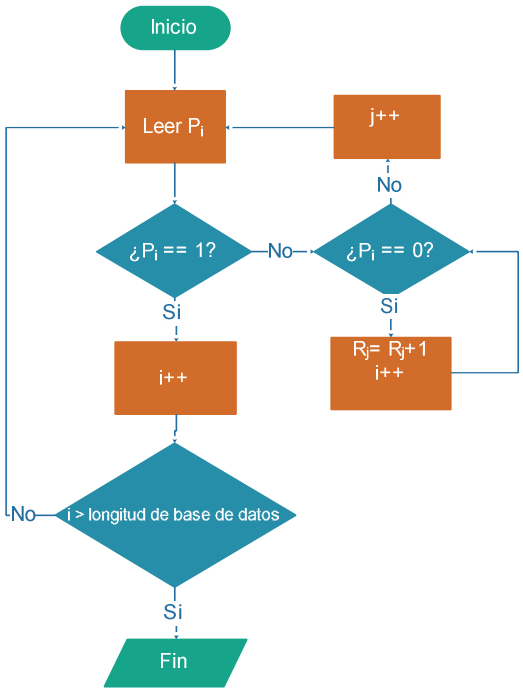


Imagen 5. 5 Cálculo de ráfagas

5.5. Análisis de jitter

El jitter puede degradar la calidad de una transmisión de audio/video, casi tanto como la pérdida de paquetes. También puede ser útil para brindar una segunda medición sobre la congestión de la red. Mientras que la pérdida de paquetes puede rastrear una congestión continua, el jitter nos permite rastrear congestiones transitorias. En presencia de jitter los tiempos entre llegadas de las tramas variarán, como se observa en la Imagen 5.6, la tercera trama (E2) llega tarde en el receptor (R2). En este escenario, el usuario podría presenciar el congelado momentáneo de imagen de la trama más recientemente entregada (R1) hasta que la trama atrasada (R2) llegue. La trama tres (R2) sería entonces reproducida brevemente con el fin de preservar el tiempo para la siguiente trama (R3)

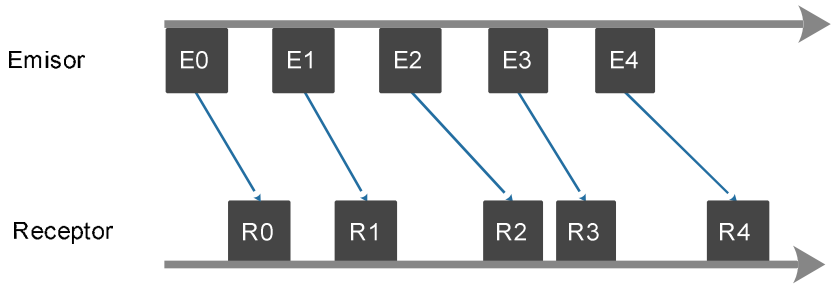
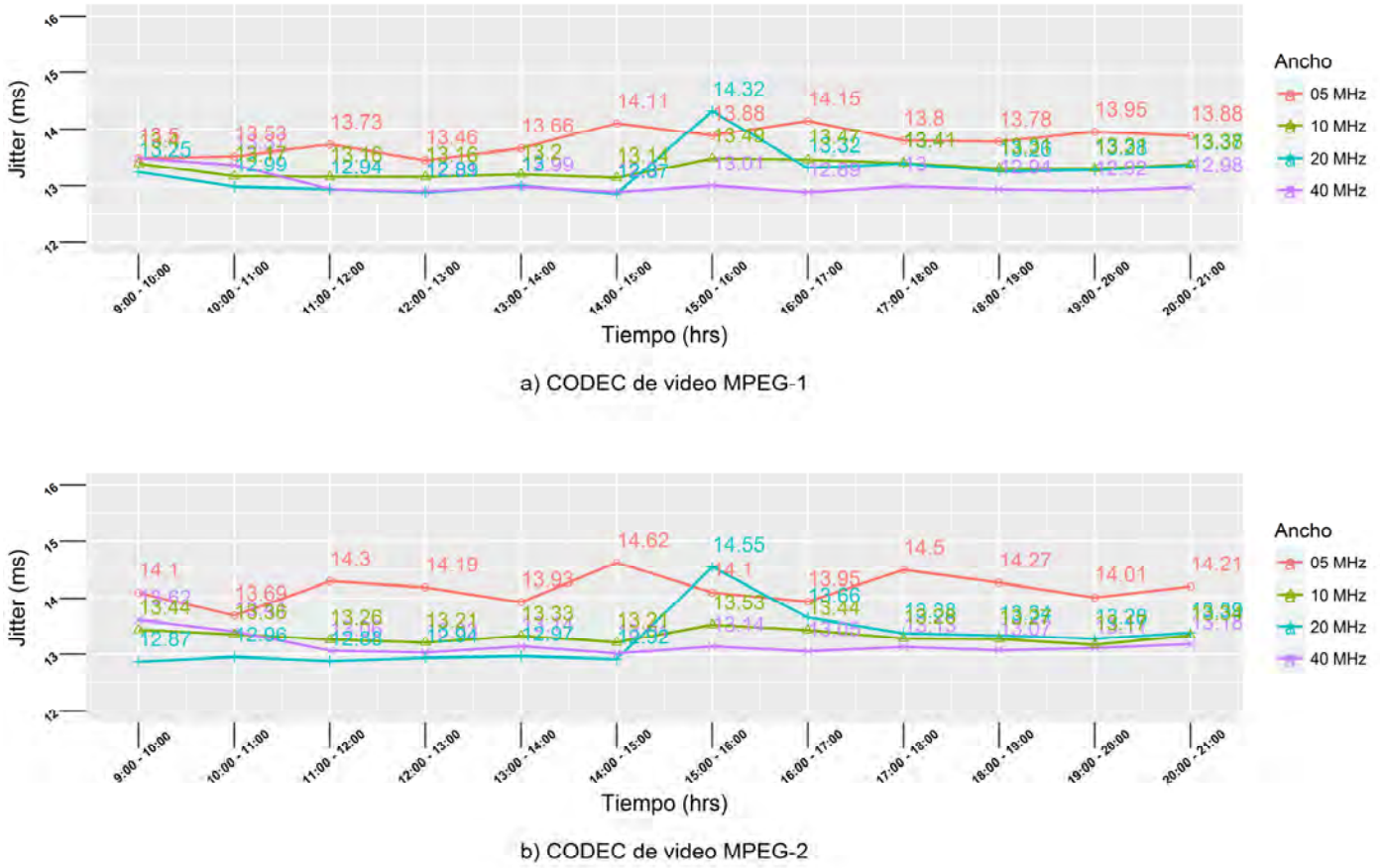


Imagen 5. 6 Efecto jitter

Si bien el uso de buffers ayuda en gran medida a solventar los efectos del jitter, es necesario elegir los tamaños correctos, ya que, un buffer grande podría aumentar el efecto de los retardos, mientras que un buffer pequeño incrementaría las pérdidas totales de paquetes, debido al desbordamiento de buffer.

5.5.1. Video



a) CODEC de video MPEG-1

b) CODEC de video MPEG-2

Imagen 5. 7 Jitter promedio - video MPEG 1 & MPEG 2

La Imagen 5.7, nos presenta el jitter promedio por hora que tuvo cada uno de los CODECs de video utilizado en este trabajo. Analizándolo por hora, podemos ver un desempeño esperado entre las dos transmisiones de video, es decir, entre mayor ancho de banda disponible, los valores del jitter son menores en las transmisiones; o dicho de otra forma, mientras menor es el ancho de banda disponible, los valores de jitter tienden a aumentar.

Enfocándonos en los CODEC de video, se observa que, por lo general, MPEG-1 fue quien tuvo valores ligeramente menores de jitter.

5.5.2. Audio

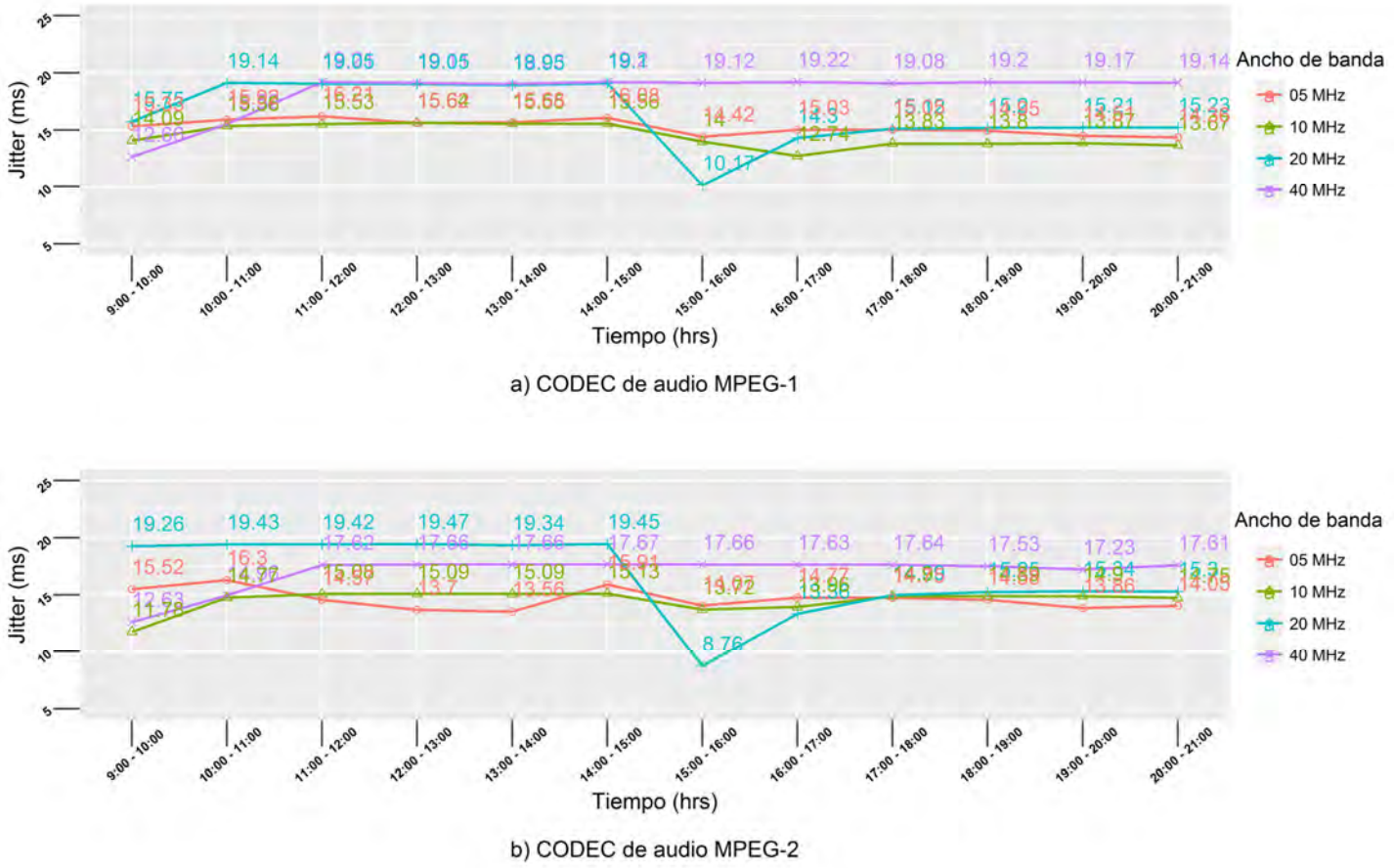


Imagen 5. 8 Jitter promedio – audio MPEG-1 & MPEG-2

La Imagen 5.8 muestra el jitter promedio por hora de los flujos correspondientes a los CODECs de audio. A diferencia de los flujos de video, en los de audio no se presenta una tendencia marcada en función de los anchos de banda. Por ejemplo, en el ancho de banda más bajo, 5 MHz, los valores de jitter se mantienen por debajo de los presentados en el ancho de banda de 40 MHz.

Observando el comportamiento de los CODECs, se observa que en general, MPEG-2 tuvo valores de jitter más bajos que MPEG-1, pero al igual que en los flujos de

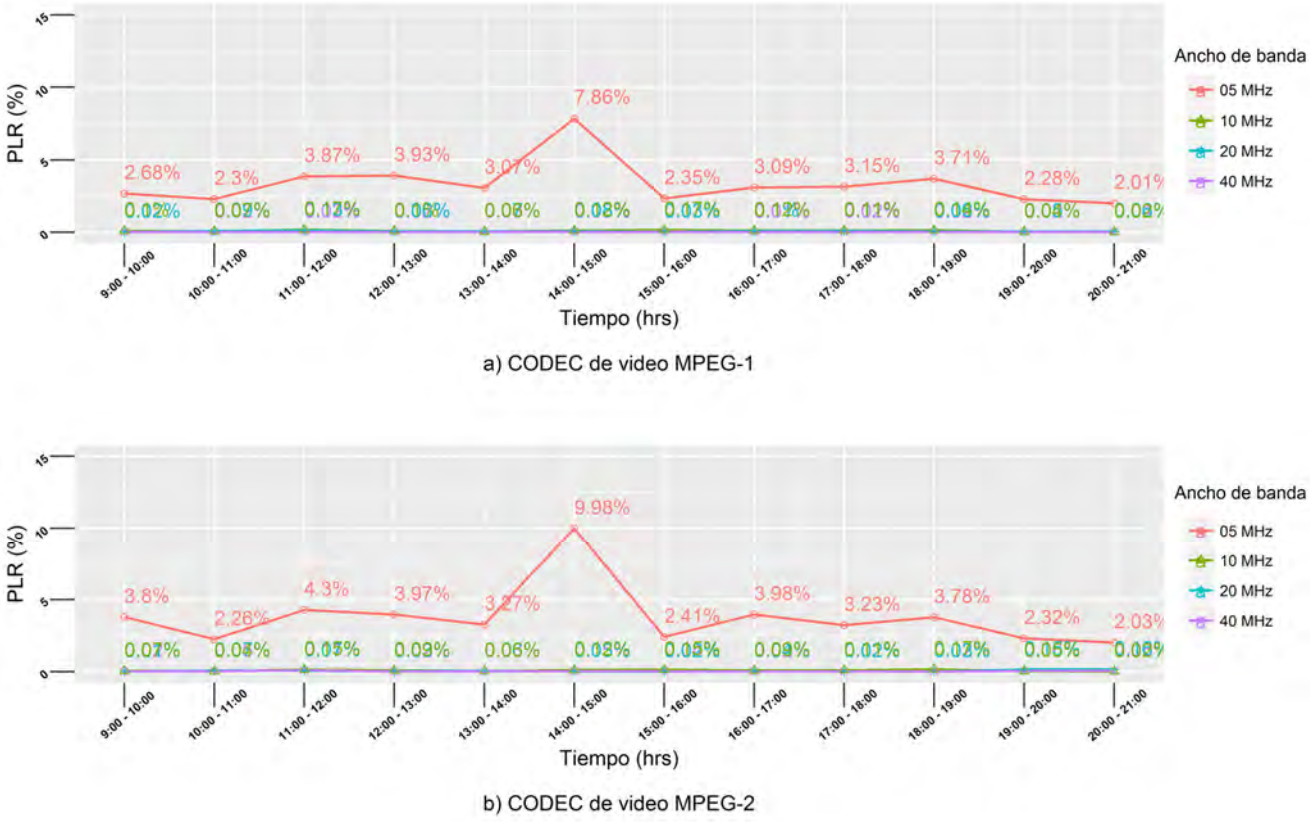
video, los dos CODECs no presentan un gran margen de diferencia entre sus valores de jitter.

Derivado de este análisis, se puede concluir que el jitter en el tráfico de video es más sensible al ancho de banda disponible a diferencia del tráfico de audio.

5.6. Análisis de pérdida de paquetes (PLR)

La pérdida de paquetes impacta significativamente en la calidad de una transmisión, dado que, al presentarse pérdidas, algunas tramas con secuencias de video no llegan al receptor. Como consecuencia de esto, el espectador presenciaría eventos tales como la congelación de la imagen del video previamente recibida, y posteriormente, observaría un salto de esta imagen hacia la siguiente recibida correctamente, creando un vacío y una degradación de la calidad de la transmisión.

5.6.1. Video



a) CODEC de video MPEG-1

b) CODEC de video MPEG-2

Imagen 5. 9 Pérdidas por hora – video MPEG-1 & MPEG-2

En la Imagen 5.9 se aprecia el porcentaje de pérdidas por hora que presentó cada CODEC de video. A primera vista, se puede apreciar el comportamiento dado en el ancho de banda más bajo, por ser el que más pérdidas tuvo, en comparación con los otros anchos de banda más grandes.

A nivel de CODEC, se observa que MPEG-1 tuvo menores porcentajes de pérdidas en contraste con MPEG-2.

5.6.2. Audio

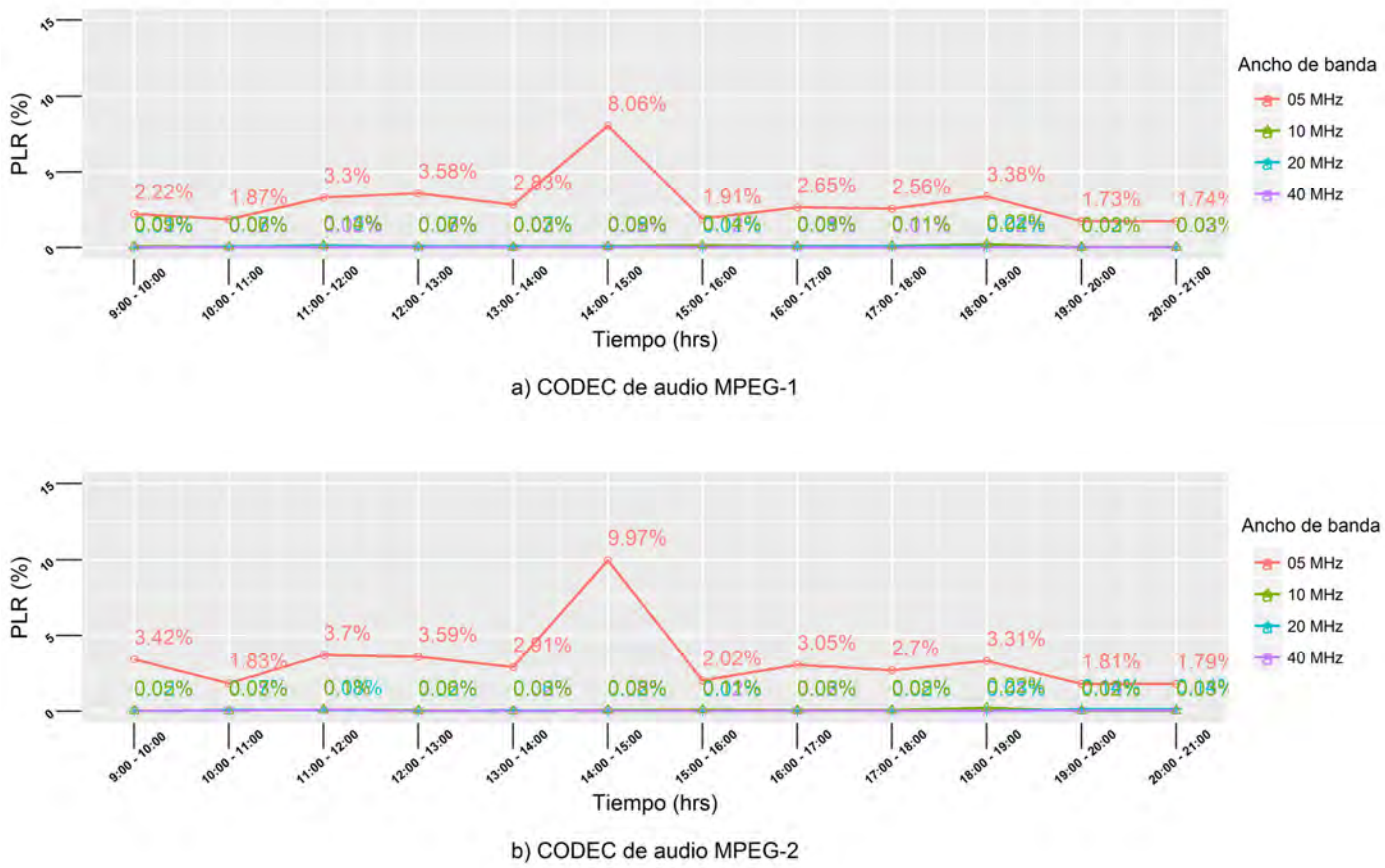


Imagen 5. 10 Pérdidas por hora – audio MPEG-1 & MPEG-2

La Imagen 5.10 describe el comportamiento de la pérdida de paquetes en los flujos de audio. Se puede observar que tanto audio y video tuvieron un comportamiento parecido, es decir, las pérdidas más considerables se dan en el ancho de banda más bajo (5MHz).

A nivel de CODEC, MPEG-1 vuelve a tener un mejor desempeño en comparación de MPEG-2.

Derivado de este análisis, se puede concluir que la pérdida de paquetes en el tráfico de video y audio, está directamente relacionada con ancho de banda disponible.

5.7. Análisis de la relación entre el jitter, pérdida de paquetes, "bursts" y "gaps"

Tener una comprensión de los efectos del jitter y la pérdida de paquetes puede ser de ayuda para mejorar la calidad perceptual de las transmisiones de video. A continuación, analizaremos la relación entre el jitter y la pérdida de paquetes.

Para llevar a cabo este análisis, nos enfocaremos en el ancho de banda de 5 MHz, dado que es donde se presentaron altos porcentajes de pérdida de paquetes.

Otros recursos que utilizaremos para realizar dicho análisis será un grupo de gráficas entre las cuales se encuentran:

a) Porcentaje de pérdida de paquetes por hora
b) Jitter promedio por hora
c) Tamaño máximo de los "bursts" por hora
d) Jitter máximo por hora

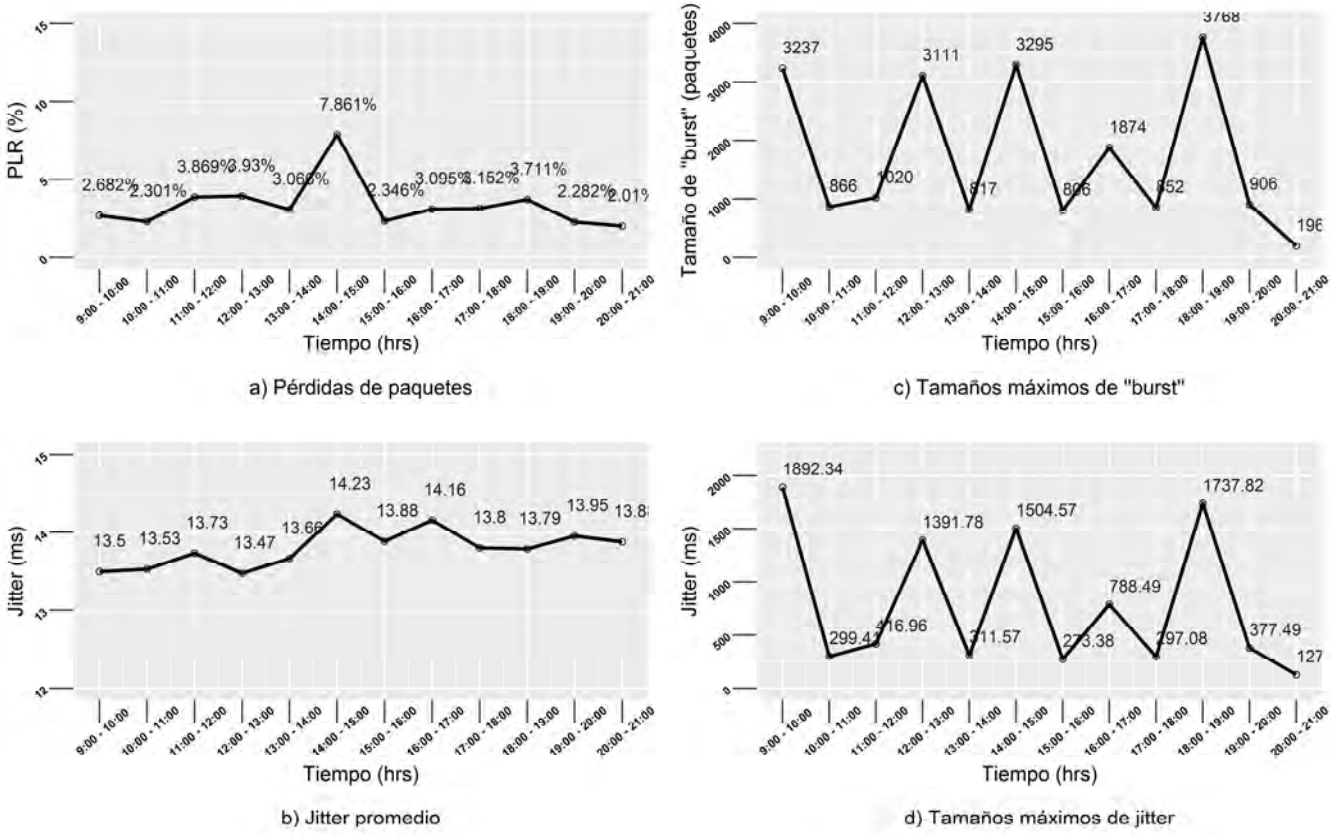


Imagen 5. 11 Análisis pérdida-jitter – video MPEG-1

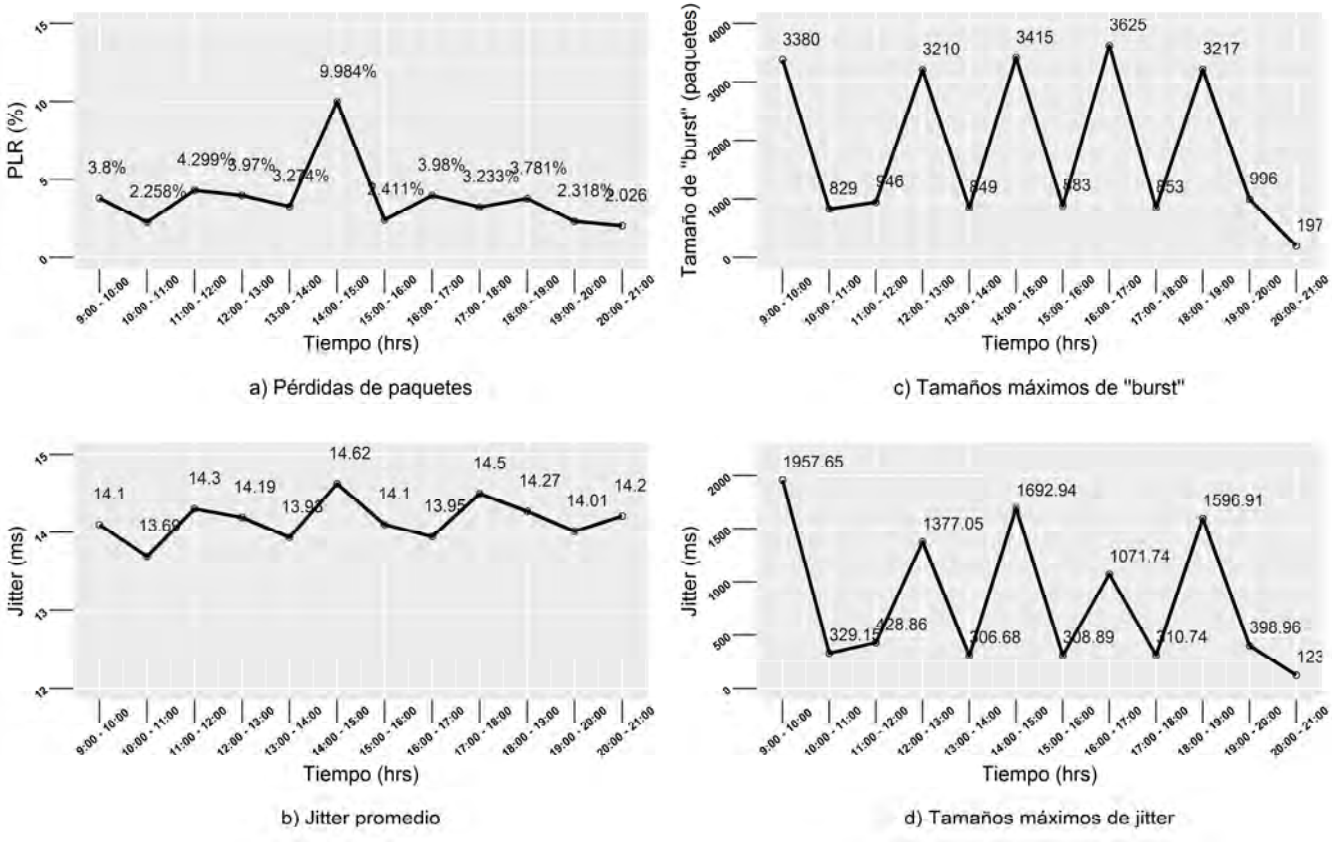


Imagen 5. 12 Análisis pérdida-jitter – video MPEG-2

En la Imagen 5.11 a) se observa que el valor máximo de PLR se presenta entre las horas 14:00 a 15:00, con un valor de 7.861%, y de igual forma se puede apreciar en la Imagen 5.11 b) que en ese mismo intervalo de tiempo se presentó el valor promedio mayor de jitter.

En la Imagen 5.11 c) se muestran los valores máximos de los “bursts” ocurridos por cada hora de medición, en contraste con la Imagen 5.11 d) se puede observar el efecto de los “bursts” sobre los valores máximos de jitter, es decir, la ocurrencia de ráfagas grandes de pérdida de paquetes puede generar valores atípicos de jitter, con impactos negativos en la calidad de la transmisión de video.

Por otro lado, en las Imágenes 5.11 a) y c) se puede apreciar que los valores promedio de pérdida de paquetes no necesariamente reflejan la ocurrencia de ráfagas grandes. Por ejemplo, el tamaño de la ráfaga más grande ocurrió entre las horas 18:00 a 19:00 (Imagen 5.11 c)) y el valor máximo de PLR se presenta entre las horas 14:00 a 15:00 (Imagen 5.11 a)). Por consiguiente, cuando analizamos el impacto de la pérdida de paquetes en una transmisión de video, es importante realizar un estudio de cómo se presentó el PLR (a ráfagas o de forma aleatoria) para poder evaluar de forma más precisa la calidad percibida por el usuario final.

Después de este análisis se puede concluir que existe una relación entre el PLR y los valores promedio de jitter, así también entre la ocurrencia de “bursts” y valores atípicos de jitter.

5.7.1. “Bursts”

Piense en dos transmisiones de 10,000 paquetes de video cada una, en donde hubo pérdida de paquetes del 1%. El video “A” pierde uno de cada 100 paquetes a través de toda la transmisión (pérdida aleatoria), mientras que el video “B” pierde 50 paquetes de forma consecutiva al principio y 50 paquetes de forma consecutiva al final de la transmisión (pérdida en ráfagas). ¿Qué video tendría mejor calidad? Para poder contestar esta pregunta, es importante analizar no sólo el porcentaje de

pérdida de paquetes, sino también la forma como se presentaron las pérdidas durante la transmisión.

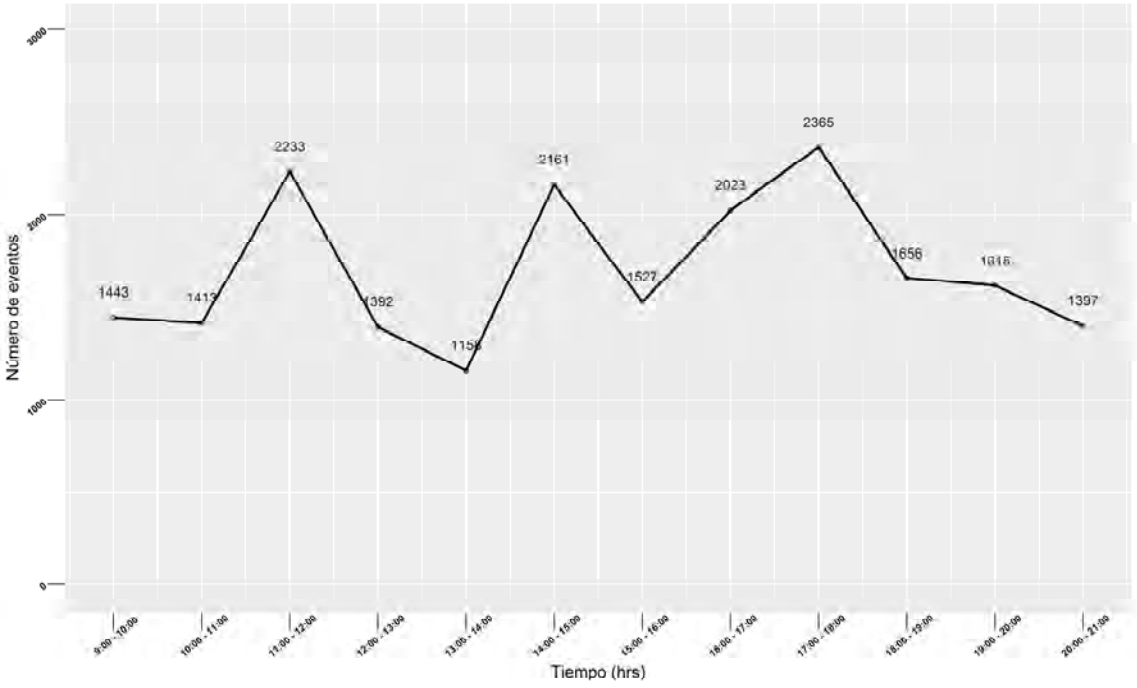


Imagen 5. 13 Eventos de pérdidas – video MPEG-1

Por ejemplo, en la Imagen 5.13 se observa que entre las horas 17:00 a 18:00 se presentó la mayor cantidad de eventos de pérdidas o “bursts” con un valor de 2365; si bien, este valor es un punto de referencia para alertarnos sobre un problema de calidad en la transmisión de video, no necesariamente un número alto de eventos de pérdidas puede lograr una mayor degradación en la calidad de una transmisión. Para tener un criterio más real, es necesario observar las características de cada uno de estos eventos, tales como: la frecuencia y el tamaño.

Observemos a continuación el comportamiento de los “bursts” y los “gaps” presentes en las horas 14:00-15:00, 17:00-18:00 y 18:00-19:00 representados en las Imágenes 5.14, 5.15 y 5.16 respectivamente.

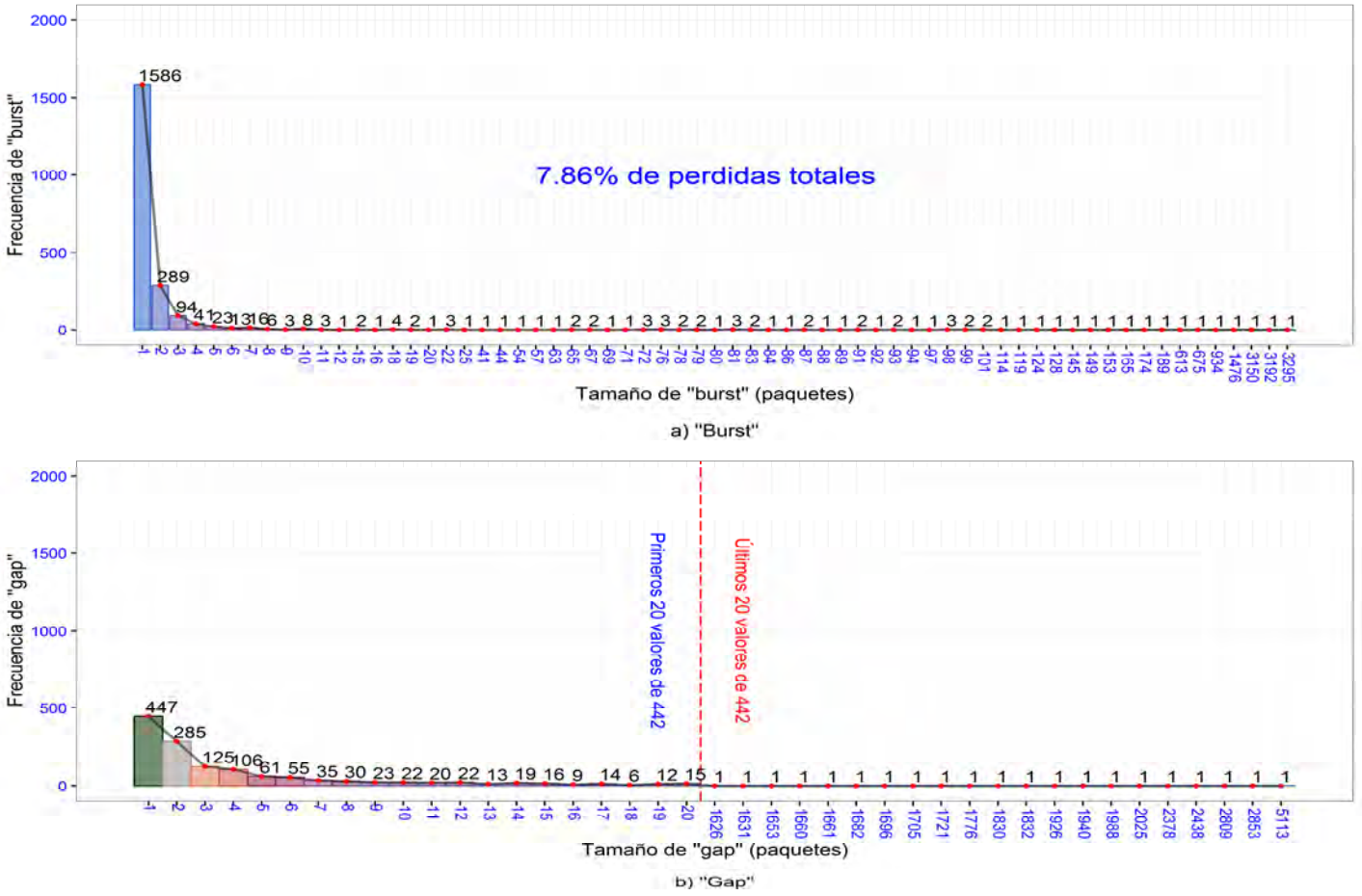


Imagen 5. 14 "Bursts" y "gaps" - 14:00-15:00 – video MPEG-1

En la Imagen 5.14 a) podemos ver que la gráfica de "bursts" presenta un decaimiento rápido, debido a que el primer valor de 65 valores del histograma, corresponde a un valor de frecuencia grande, equivalente a 1586 ("burst" de tamaño de 1 paquete perdido), con respecto al segundo valor de frecuencia de 289; por otro lado, al final de la cola se van presentando "bursts" de gran tamaño, pero con baja frecuencia, de los cuales los 3 últimos tienen un tamaño promedio de 3000 paquetes perdidos, rango bastante elevado que impacta seriamente en la calidad perceptual de la transmisión de video. Estos valores grandes de "bursts" se ven reflejados en el valor atípico de jitter de 1504.57ms presentado en la Imagen 5.11d) entre las horas 14:00 a 15:00.

En la Imagen 5.14 b) se muestra que la gráfica de "gaps" presenta un decaimiento más lento respecto a la gráfica de "bursts", debido a que la diferencia entre el primer valor de frecuencia de 442 valores del histograma y los siguientes no presentan una diferencia grande. Por otro lado, el hecho de que el "gap" de tamaño más grande

sea de 5113 paquetes de 279242 recibidos puede indicar constantes degradaciones en la calidad perceptual de la transmisión.

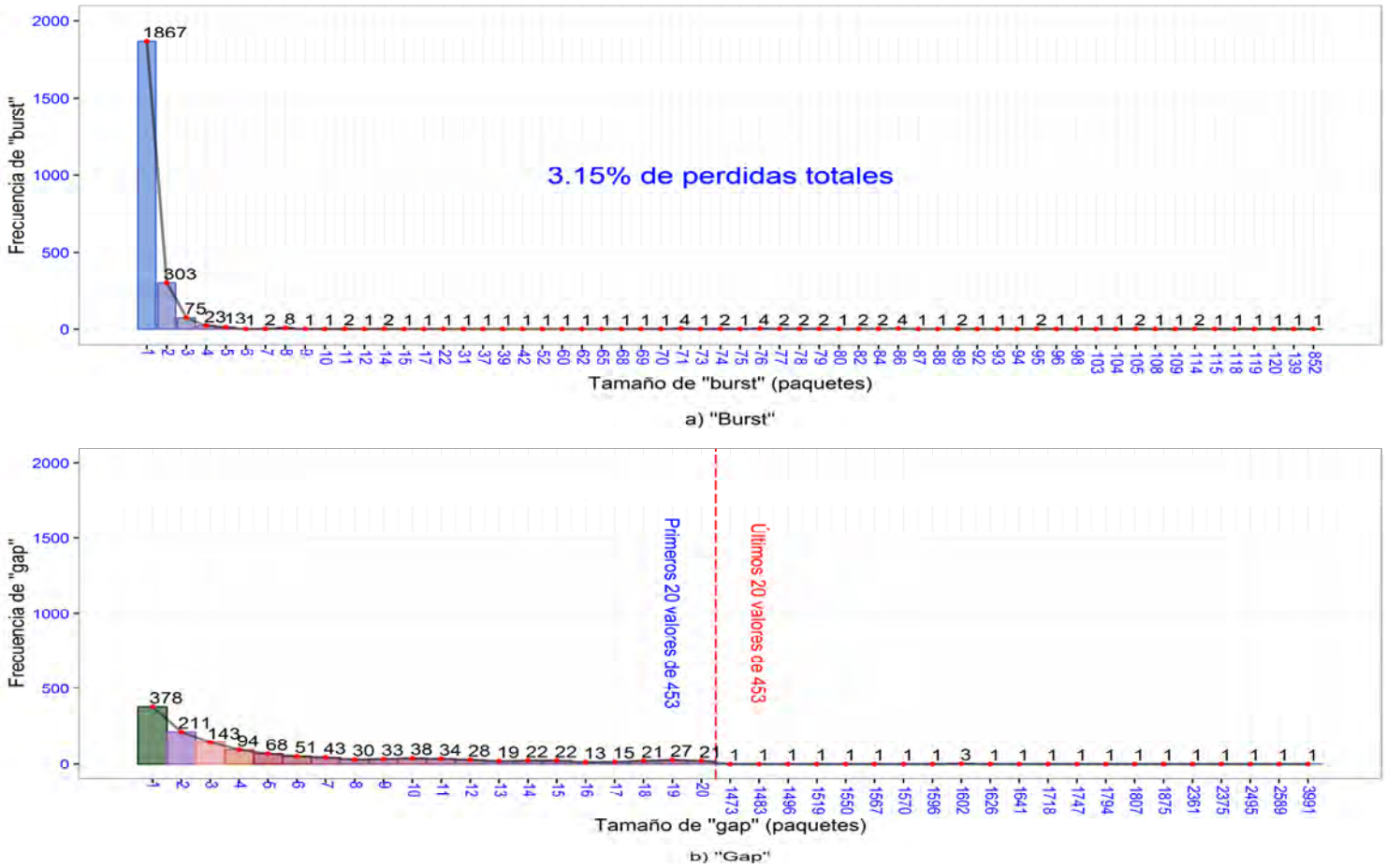


Imagen 5. 15 “Bursts” y “gaps” - 17:00-18:00 – video MPEG-1

La Imagen 5.15 muestra un comportamiento similar a la Imagen 5.14 respecto al decaimiento de los histogramas de “bursts” y “gaps”, sin embargo, se puede observar en la Imagen 5.15 a), que ningún tamaño de “burst” supera los 1000 paquetes perdidos de manera consecutiva, como se presentó en la Imagen 5.14 a). Este hecho, podemos verlo reflejado en la Imagen 5.11 d) en el tamaño de jitter máximo que se presentó en la medición de las 17:00 a 18:00 horas, con un valor de 297.08 ms. Con respecto a los “gaps”, podemos observar que el tamaño mayor de gap fue de 3991 paquetes (menor que en la medición de las 14:00 a 15:00 horas), lo cual indica que existió mayor presencia de eventos de pérdidas durante esta hora, pero con ráfagas de menor tamaño.

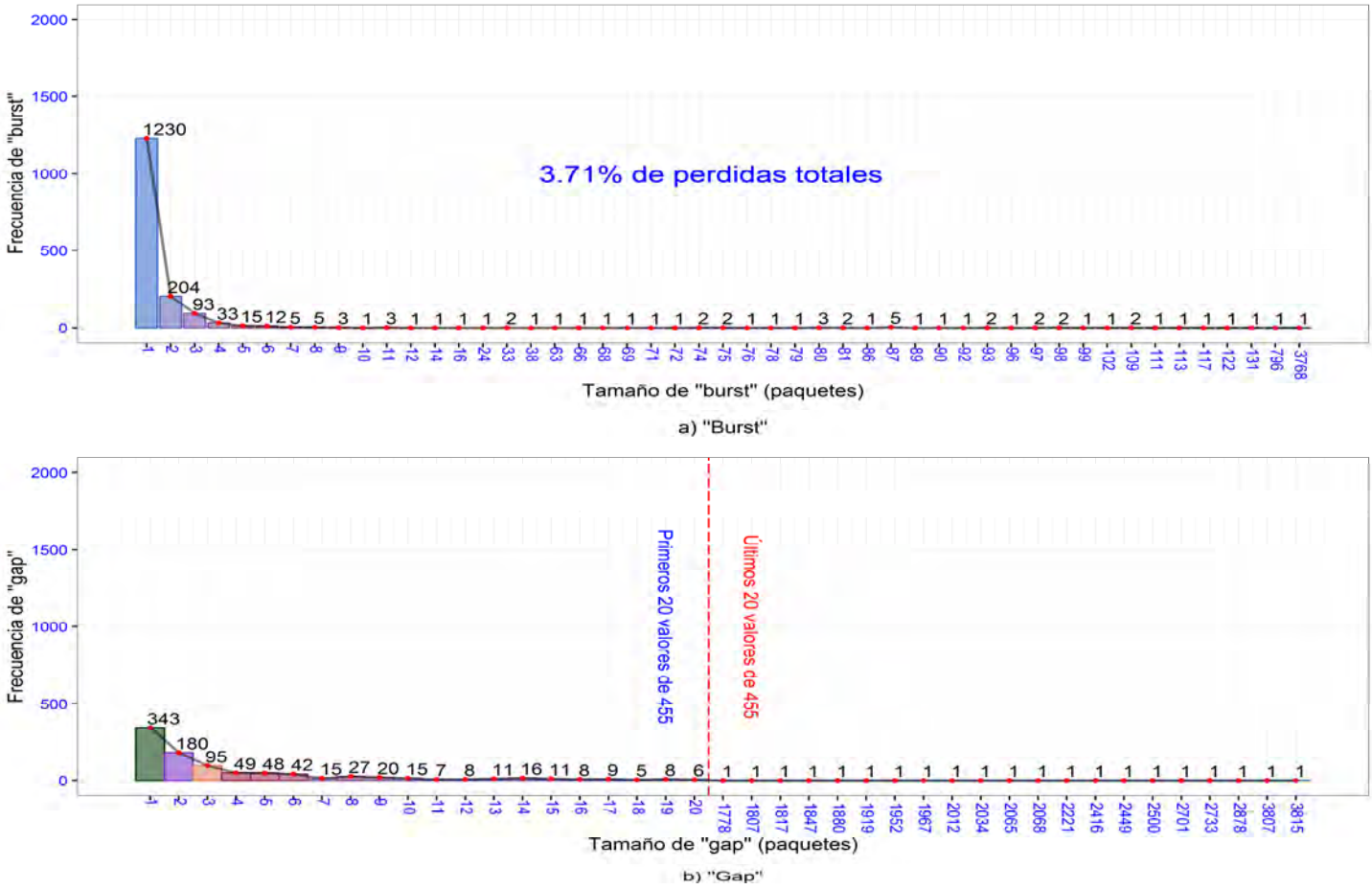


Imagen 5. 16 "Bursts" y "gaps" - 18:00-19:00 – video MPEG-1

Al igual que en las dos graficas previamente analizadas, observamos tanto en la gráfica de ráfagas, como en la de gaps, la presencia de largas colas. Pero comparando los valores máximos de frecuencia en las gráficas de bursts y gaps, vemos que esta hora tiene menor tamaño que las dos anteriores, lo cual es producto de un menor número de eventos de pérdidas, teniendo esta una cantidad de 1656 eventos, mientras que las otras dos horas superan los 2000 eventos.

Por otro lado, esta hora fue la que tuvo un jitter máximo superior a las 2 anteriores (1737.82 ms), esto es derivado de una ráfaga de 3768 paquetes. Sin embargo, solo hubo 1 ráfaga superior a los 1000 paquetes perdidos, a diferencia de la hora 14:00-15:00 donde hubieron 3 ráfagas superiores a los 1000 paquetes. Lo que puede verse reflejado en el jitter promedio obtenido en esta hora, con valor de 13.79 ms, en contraste con los 14.23 ms de jitter promedio obtenido en la hora 14:00-15:00, a consecuencia de una congestión mayor en la red.

La gráfica de gaps, de nuevo presenta una larga cola de 455 valores, llegando a un total de 455 gaps durante la hora y con un tamaño máximo de gap de 3815 paquetes, muy parecido a la hora anterior.

5.7.2. Relación entre las pérdidas de paquetes y el jitter

Con el objetivo de realizar un análisis más profundo, referente a la relación encontrada entre la pérdida de paquetes y el jitter, presentamos las Imágenes 5.17, 5.18 y 5.19 en las cuales, se podrá observar las ráfagas de las 3 horas descritas anteriormente (14:00-15:00, 17:00-18:00 y 18:00-19:00).

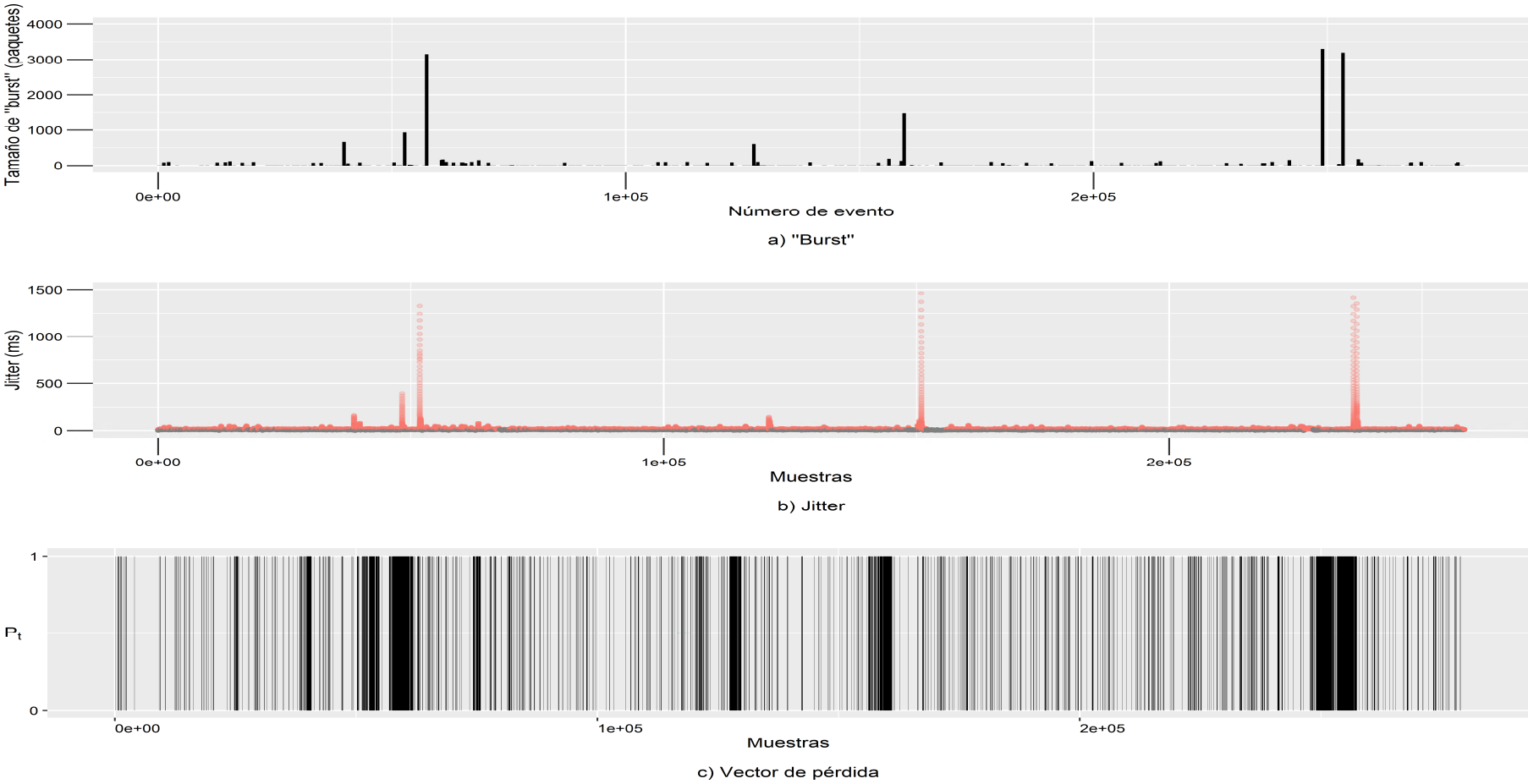


Imagen 5. 17 Relación perdidas-jitter - 14:00-15:00 – video MPEG-1

La Imagen 5.17 a) muestra los tamaños de cada uno de los eventos de bursts presentados durante la transmisión. La Imagen 5.17 b) ilustra los valores de jitter presentados en cada instante de tiempo durante la transmisión. La Imagen 5.17 c) presenta la gráfica del vector de pérdida de la transmisión. Un vector de pérdida estará representado por una secuencia binaria $X = \{X_t : t = 1, \dots, N\}$, donde, $X_t = 1$ quiere decir que el paquete fue perdido y $X_t = 0$ que el paquete fue recibido o llegó a su destino. Las Imágenes 5.17 a), 5.17 b) y 5.17 c), permiten observar el comportamiento temporal de la pérdida de paquetes y el jitter; y de igual forma, la relación existente entre el jitter y la pérdida de paquetes. Se puede observar en la Imagen 5.17 a), cuando se presentan ráfagas con tamaños considerables, los valores del jitter se incrementan. Lo interesante de este estudio es que muestra el comportamiento temporal de cada uno de los deterioros, es decir, permite ver en qué momento de la transmisión ocurren. Por ejemplo, en la hora de 14:00-15:00, podemos ver que al inicio de la transmisión (~10min) se presentó una ráfaga de aproximadamente 900 paquetes perdidos de manera consecutiva, y a los pocos minutos se presentaron dos ráfagas más, una de aproximadamente 1000 paquetes y otra de más de 3000 paquetes; seguido de una agrupación de ráfagas con menor tamaño en la parte media de la transmisión, y dos mayores a 3000 paquetes al final de la transmisión. Todas las ráfagas de paquetes perdidos mencionadas anteriormente, impactaron directamente en la aparición de valores atípicos de jitter como se puede ver en la Imagen 5.17 b).

Por otro lado, en la hora 17:00-18:00 se puede observar principalmente una ráfaga de aproximadamente 900 paquetes perdidos de manera consecutiva y en consecuencia su correspondiente valor atípico de jitter con un valor mayor a los 250ms.

Finalmente, en la hora 18:00-19:00, se observan dos ráfagas principales, una de aproximadamente 900 paquetes y otra de más de 3500 paquetes, de igual forma como en las imágenes anteriores, se puede apreciar el efecto de estas ráfagas, con la aparición de sus correspondientes valores atípicos de jitter.

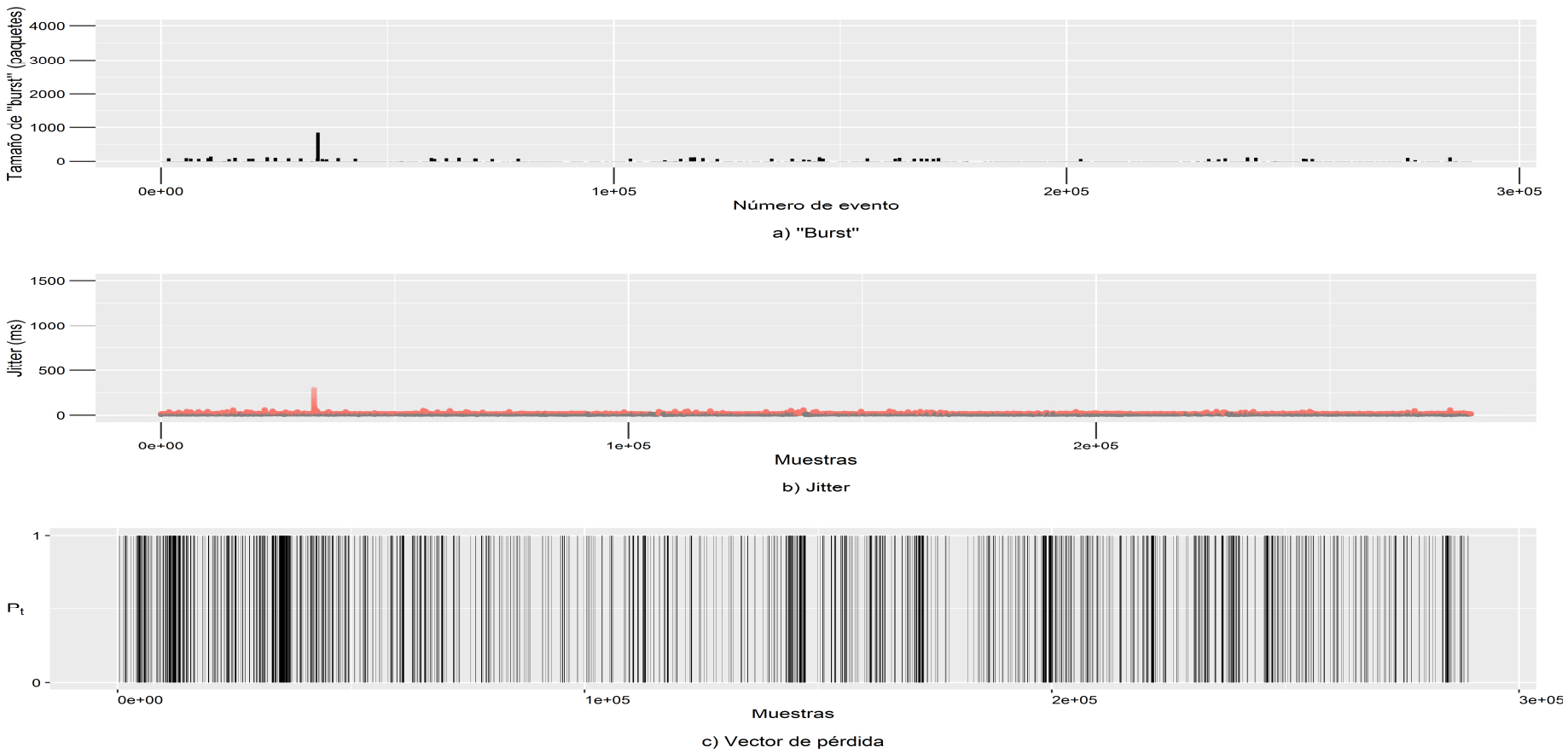


Imagen 5. 18 Relación perdidas-jitter - 17:00-18:00 – video MPEG-1

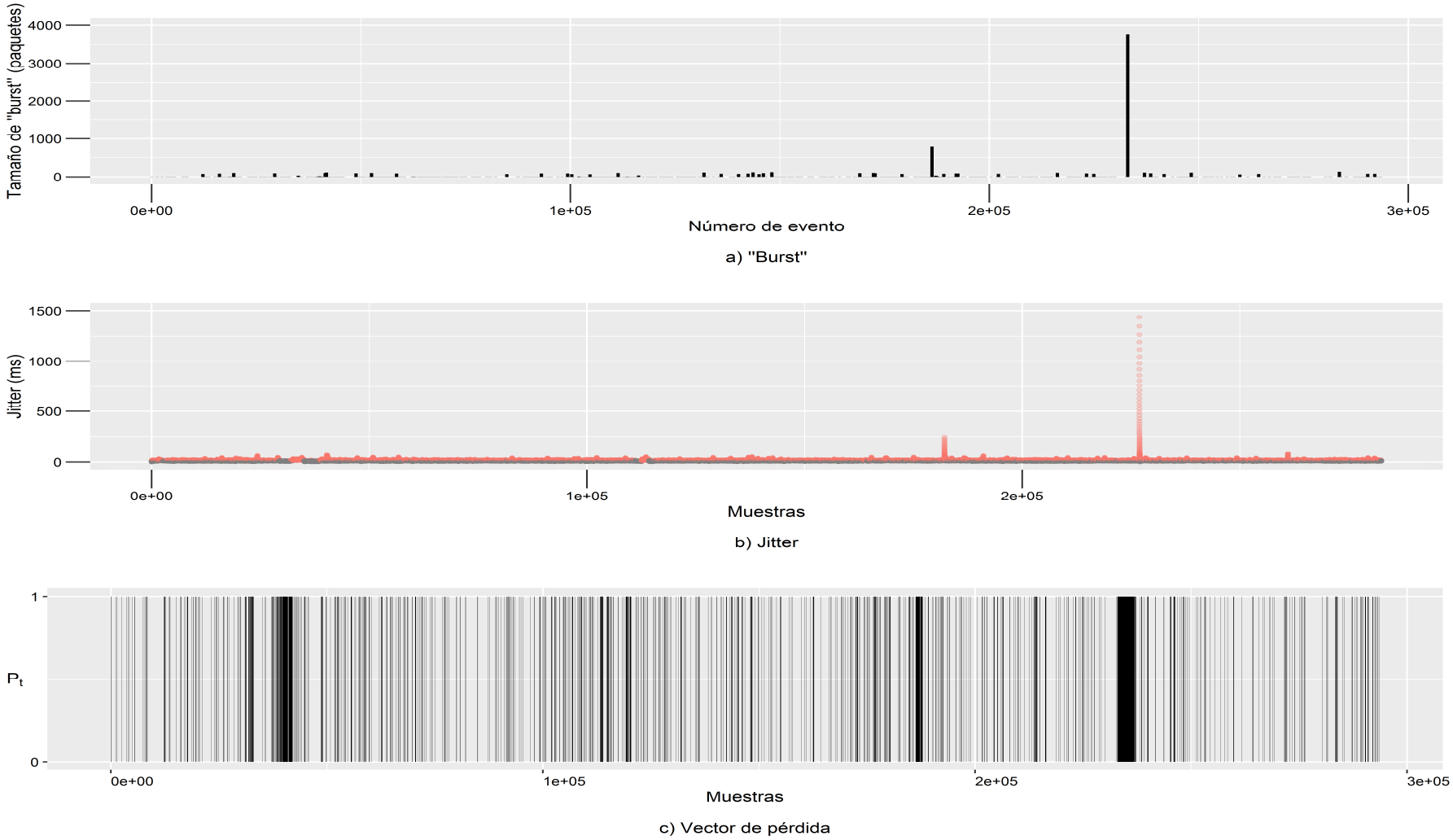


Imagen 5. 19 Relación perdidas-jitter - 18:00-19:00 – video MPEG-1

5.7.1. MPEG-1 vs MPEG-2

En los apartados anteriores presentamos un análisis de la relación entre la pérdida de paquetes y el jitter en una transmisión de video streaming mediante el CODEC MPEG-1, sobre un enlace de 5 MHz; en esta sección, presentaremos un análisis del comportamiento de los CODECs MPEG-1 y MPEG-2, y un análisis comparativo entre ambos CODECs con el objetivo de contrastar su desempeño, transmitiendo de forma paralela.

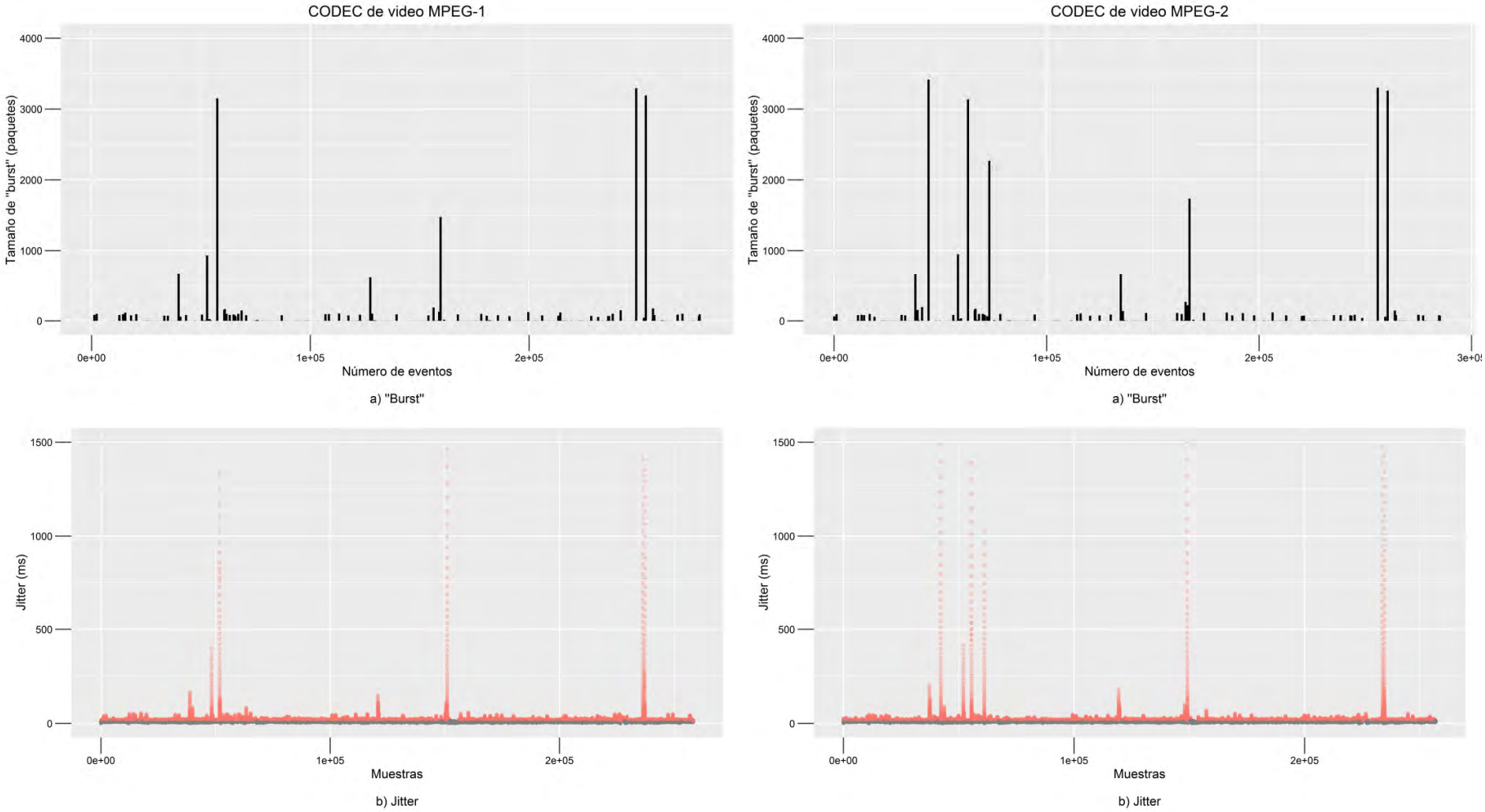


Imagen 5. 20 Bursts vs jitter - 14:00-15:00 – video MPEG-1 / MPEG-2

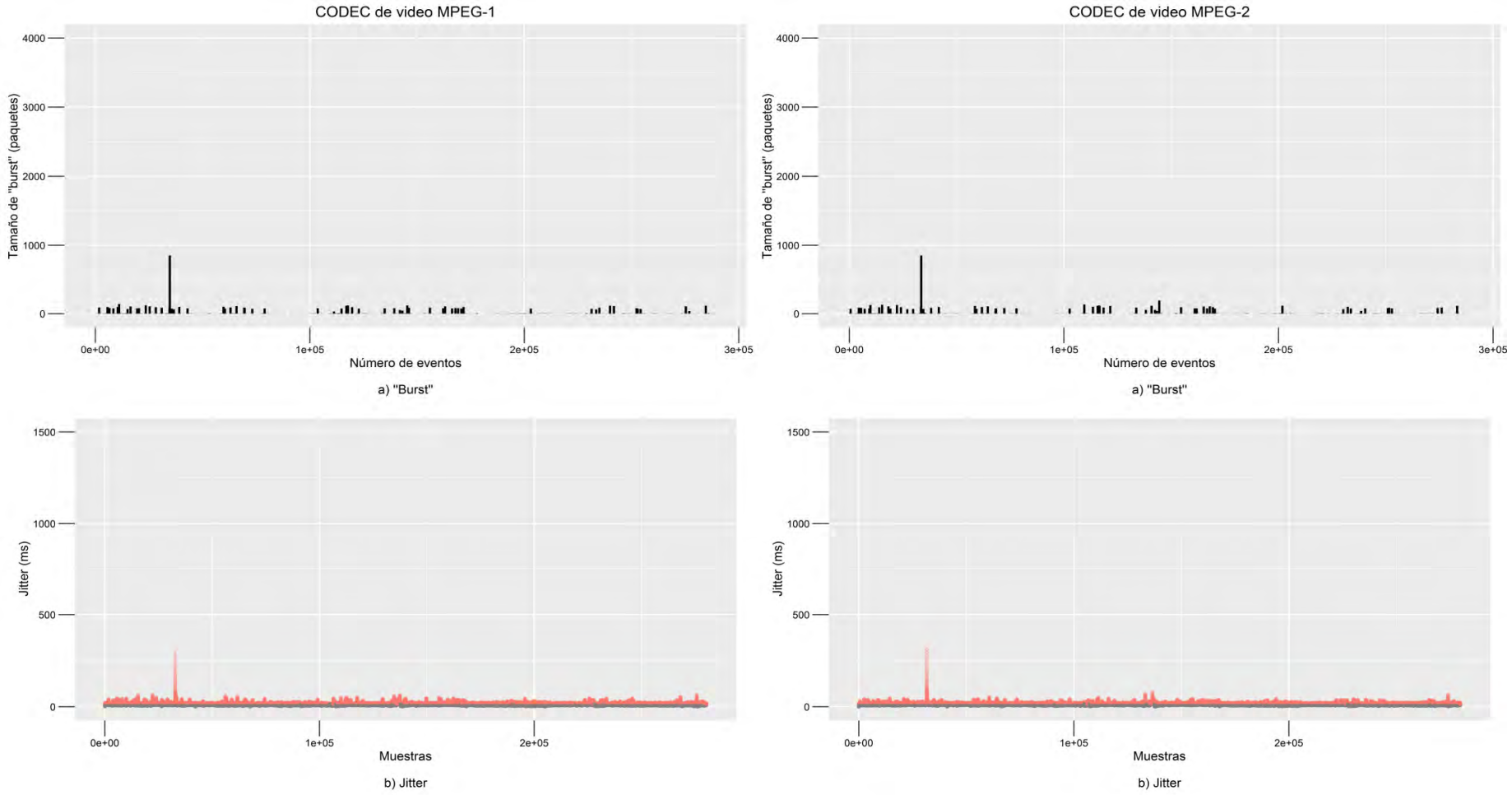


Imagen 5. 21 Bursts vs jitter - 17:00-18:00 – video MPEG-1 / MPEG-2

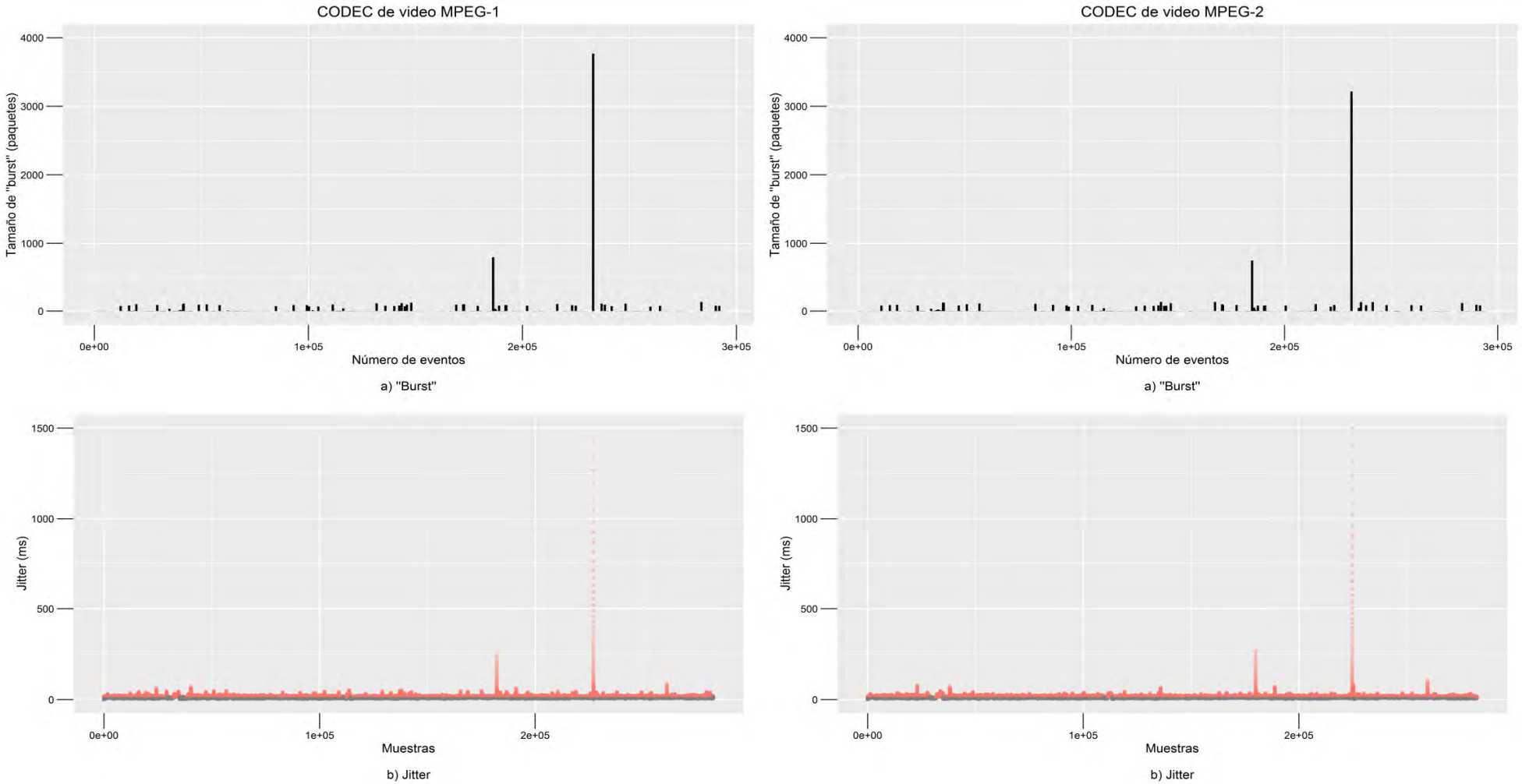


Imagen 5. 22 Bursts vs jitter - 18:00-19:00 – video MPEG-1 / MPEG-2

En las Imágenes 5.20, 5.21 y 5.22 podemos observar el comportamiento temporal de los “bursts” y el jitter en los CODECs MPEG-1 y MPEG-2; y de igual forma, la relación existente entre el jitter y la pérdida de paquetes a ráfagas. Se puede visualizar que cada vez que se presentan ráfagas con tamaños considerables, los valores del jitter se incrementan en ambos CODECs dando lugar a la aparición de valores atípicos de jitter.

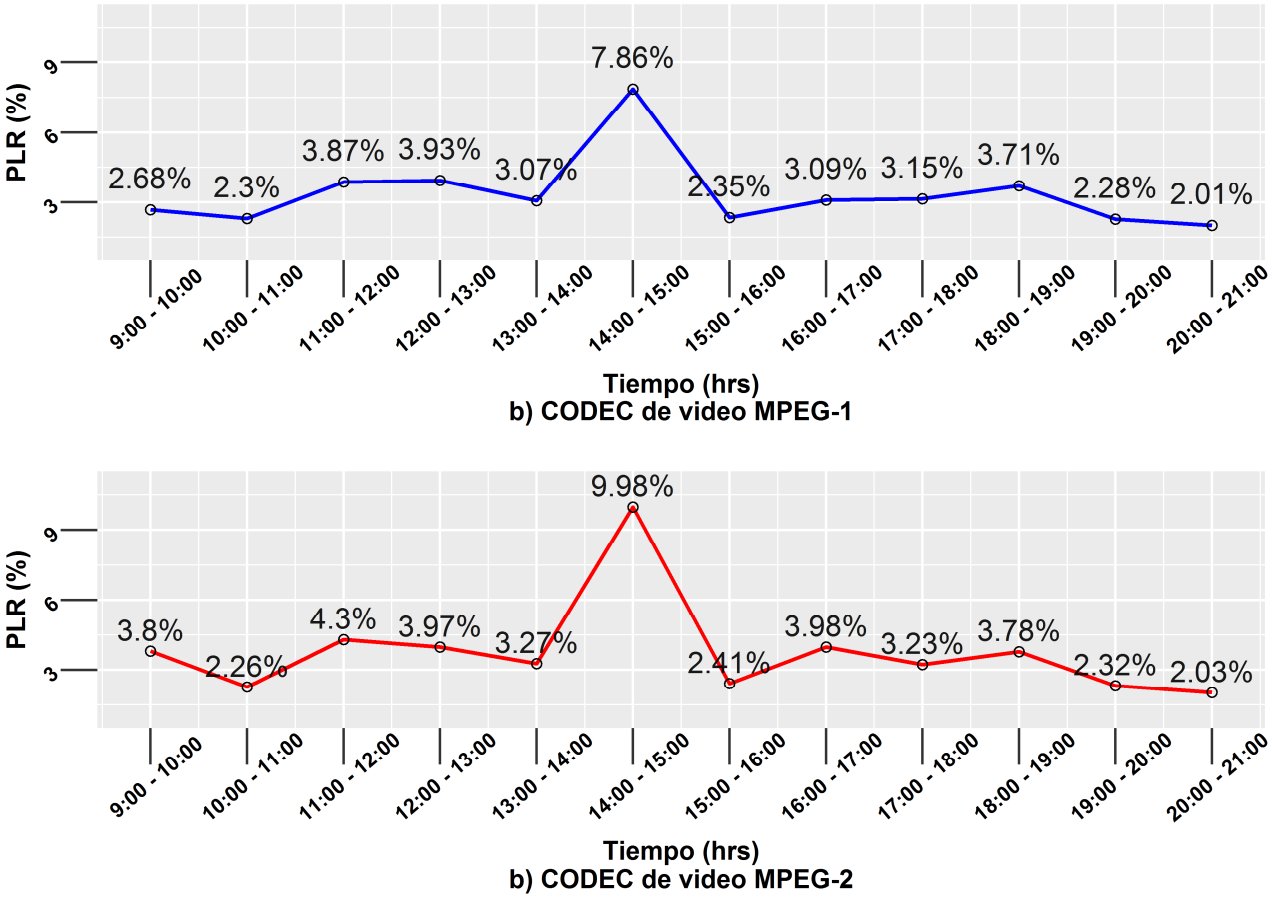


Imagen 5. 23 Pérdidas de paquete – video MPEG-1 / MPEG-2

La imagen 5.23 muestra los diversos valores de PLR que se presentaron durante 12 horas de medición para los CODECs MPEG-1 y MPEG-2, como se puede observar, el CODEC MPEG-2 es más sensible a las pérdidas de paquetes, es decir en la transmisión de video con MPEG-2, se presentaron porcentajes mayores de PLR.

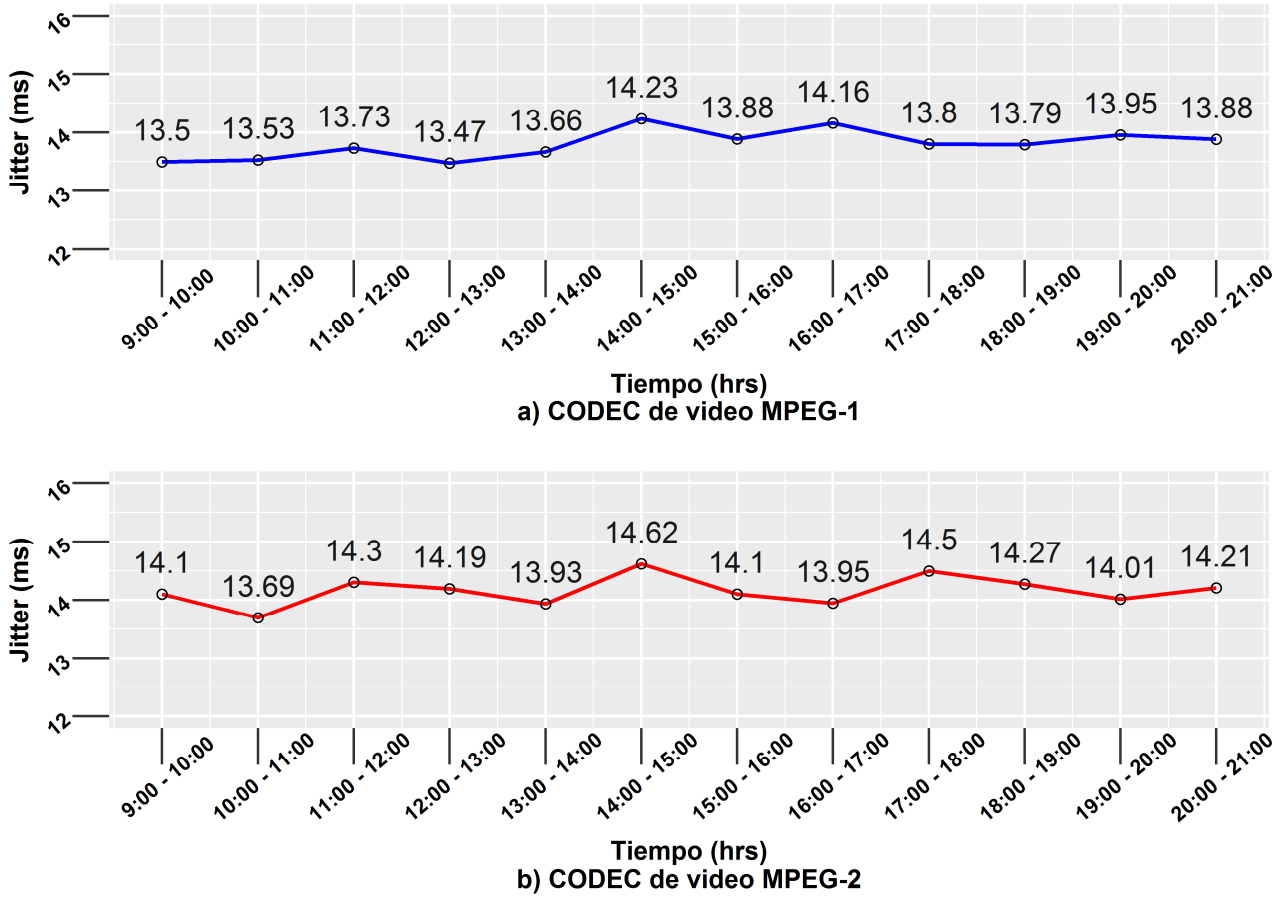


Imagen 5. 24 Jitter – video MPEG-1 / MPEG-2

Por otro lado, la imagen 5.24 presenta los valores de jitter que se presentaron durante 12 horas de medición para los CODECs MPEG-1 y MPEG-2, como se puede observar, el CODEC MPEG-2 es más sensible al jitter, es decir en la transmisión de video con MPEG-2, se presentaron valores mayores de jitter.

CAPÍTULO VI. CONCLUSIONES

El Internet de nuestros días se ha popularizado por ser un nuevo medio para la distribución de entretenimiento, así como permitir nuevos tipos de contenido multimedia, más personalizados e interactivos. En los últimos años el servicio de video streaming se ha popularizado en Internet gracias a plataformas como Youtube, Vimeo y Netflix. Por otra parte; este incremento de tráfico multimedia ha impactado principalmente en el consumo del ancho de banda y disminución de recursos disponibles en la red IP. Derivado de los puntos mencionados anteriormente, los servicios de video streaming (en directo o bajo demanda) en redes IP enfrentan reto de garantía de calidad de servicio, especialmente en redes de gran escala.

Las pérdidas de paquetes en redes IP son inevitables, y en aplicaciones de video streaming tienen un impacto considerable en otros parámetros de QoS que finalmente deterioran la calidad perceptual del video. En el presente trabajo de tesis se realizó la caracterización temporal del parámetro pérdida de paquetes y su impacto en la métrica de jitter en una transmisión de video streaming.

Para la generación de trafico streaming se utilizó la aplicación VLC, mediante la cual, se generaron de forma paralela dos flujos de video, uno bajo el CODEC MPEG-1 y el otro bajo MPEG-2, ambos transportados por del protocolo UDP sobre enlaces con anchos de banda de 40 MHz, 20 MHz, 10 MHz y 5 MHz. La captura automatizada de los paquetes se realizó con Wireshark, sin embargo, Wireshark no cuenta con un módulo especializado para el análisis de parámetros como jitter y pérdidas en ráfagas para servicios de video streaming. Dado las escazas opciones de software libre dedicadas al análisis de métricas de calidad de video streaming, se optó por la creación de 3 scripts (jitter, pérdidas y ráfagas-gaps) para el tratamiento y análisis de los flujos capturados. Dichos scripts fueron desarrollados en R por su versatilidad en la exploración de datos estadísticos y sus funcionalidades en la creación de gráficas.

Con los resultados del análisis de las gráficas podemos destacar los siguientes puntos:

- El jitter en el tráfico de video es más sensible al ancho de banda disponible a diferencia del tráfico de audio.
- La pérdida de paquetes en el tráfico de video y audio, está directamente relacionada con ancho de banda disponible.
- MPEG-1 fue el CODEC que tuvo menor porcentaje de pérdida de paquetes y valores más pequeños de jitter.
- Los resultados indican que incluso niveles relativamente altos de pérdidas de paquetes individuales no impactan drásticamente en el jitter. Sin embargo, si las pérdidas de paquetes se presentan en ráfagas, las consecuencias pueden ser negativas.
- Grandes ráfagas de perdidas impactan directamente en la aparición de valores atípicos de jitter.

La herramienta R-statistics, puede ser usada para estudiar los patrones de pérdida a partir de los números de secuencia, y extraer estadísticas como: número de paquetes perdidos, número de paquetes recibidos, paquetes recibidos fuera de orden o secuencia, número de paquetes perdidos de forma consecutiva, longitud de pérdidas consecutivas medida en número de paquetes, número de gaps entre paquetes perdidos de forma consecutiva, longitud de gaps medida en número de paquetes, etc.

La caracterización obtenida a partir de las estadísticas mencionadas anteriormente, puede ser usada en trabajos futuros para aplicar de manera correcta algún mecanismo como FEC o interleaving que contribuya a incrementar el desempeño en el servicio de video streaming.

6.1 Trabajo futuro

Las líneas futuras de trabajo pueden ser dedicados a examinar las características de las pérdidas de paquetes en ráfagas en escenarios donde se apliquen técnicas para disminuir el impacto de la pérdida de paquetes como FEC e Interleaving y evaluar comportamiento de otros parámetros QoS.

Otro punto de enfoque se basaría en observar el desempeño de nuevos CODECs de video como H.264, H.265, VP8, VP9 o MPEG-4.

Bibliografía

Whitepaper 802.11n Primer. (5 de Agosto de 2008). Obtenido de AIRMAGNET: Wireless Network Assurance: http://airmagnet.flukenetworks.com/assets/whitepaper/WP-802.11nPrimer.pdf

Apostolopoulos, J., Tan, W.-t., & Wee, S. (18 de Septiembre de 2002). *Video Streaming: Concepts, Algorithms, and Systems.* Obtenido de HP Labs: http://www.hpl.hp.com/techreports/2002/HPL-2002-260.pdf

Aramvith, S., & Sun, M.-T. (2005). MPEG-1 AND MPEG-2 Video Standards. En A. Bovik, & A. Bovik (Ed.), *Handbook of Image and Video Processing* (Segunda ed., págs. 833-847). Academic Press.

Aruba Networks . (s.f.). *Designed for Speed: Network Infrastructure in an 802.11n World.* Obtenido de Aruba White Paper: http://www.arubanetworks.com/pdf/technology/whitepapers/wp_Designed_Speed_802.11n.pdf

Aurelius, A., Lagerstedt, C., & Kihl, M. (11 de Junio de 2011). Streaming media over the Internet: Flow based analysis in live access networks. *Broadband Multimedia Systems and Broadcasting (BMSB), 2011 IEEE International Symposium on*, 1-6.

Austerberry, D. (2013). *The Technology of Video and Audio Streaming* (Segunda ed.). Estados Unidos: Focal Press.

Bai, H., Atiquzzaman, M., & Ivancic, W. (3 de Julio de 2002). Running Integrated Services over Differentiated Service Networks: Quantitative Performance Measurements. *SPIE Proceedings, 4866*.

Bellavista, P. (2009). *Telecomunication Systems and Technologies* (Vol. 2). (P. Bellavista, Ed.) EOLSS Publications.

Bonaventure, O. (30 de Octubre de 2011). *Computer Networking : Principles, Protocols and Practice.* Obtenido de The Saylor Foundation: http://www.saylor.org/site/wp-content/uploads/2012/02/Computer-Networking-Principles-Bonaventure-1-30-31-OTC1.pdf

Bucknall, J. (2012). The history of streaming media. *PCPlus 324*.

Calyam, P., Krymskiy, D., Sridharan, M., & Schopis, P. (s.f.). *Active and Passive Measurements on Campus, Regional and National Network Backbone Paths.* Obtenido de Ohio Academic Resources Network: https://www.oar.net/files/initiatives/research/PDFs/cnr_icccn05.pdf

Cerf, V. G., & Kahn, R. E. (5 de Mayo de 1974). *A Protocol for Packet Network Intercommunication.* Obtenido de Princeton University: Department of Computer Science: https://www.cs.princeton.edu/courses/archive/fall06/cos561/papers/cerf74.pdf

Chua, T.-k., & Pheanis, D. C. (Nov - Dec de 2006). QoS Evaluation of Sender-Based Loss-Recovery Techniques for VoIP. *IEEE Network*, 14-22.

Cisco. (30 de Enero de 2014). *Cisco IOS Quality of Service Solutions Configuration Guide, Release 12.2*. Obtenido de Cisco: http://www.cisco.com/c/en/us/td/docs/ios/12_2/qos/configuration/guide/fqos_c/qcfintro.html#wp1000940

Cisco. (s.f.). *The QoS Baseline.* Obtenido de Cisco Systems: https://www.cisco.com/en/US/technologies/tk543/tk759/technologies_white_paper0900aecd80295a9b.pdf

Davies, G. (2011). *Day One: Deploying Basic QoS.* (P. Ames, Ed.) Estados Unidos: Juniper Networks Books.

Fecheyr-Lippens, A. (Enero de 2010). *A Review of HTTP Live Streaming.* Obtenido de Andrew Fecheyr: http://files.andrewsblog.org/http_live_streaming.pdf

Flannagan, M., Durand, B., Sommerville, J., Buchmann, M., & Fuller, R. (2001). *Administering Cisco QoS for IP Networks.* Estados Unidos: Syngress Publishing.

Forouzan, B. (2006). *Data Communications and Networking* (Cuarta ed.). New York, Estados Unidos: McGraw-Hill.

Hackmann, G. (21 de Marzo de 2006). *802.15 Personal Area Neworks*. Obtenido de Washington University in St. Louis School of Engineering & Applied Science, Department of Computer Science & Engineering: http://www.cse.wustl.edu/~jain/cse574-06/ftp/wpans.pdf

Hanzo, L., Cherriman, P., & Streit, J. (2007). *Video Compression and Communications : From Basics to H.261, H.263, H.264, MPEG4 for DVB and HSDPA-Style* (Segunda ed.). Londres, Inglaterra: Wiley-IEEE Press.

Haskell, B., & Puri, A. (2012). MPEG Video Compression Basics. En L. Chiariglione (Ed.), *The MPEG Representation of Digital Media* (págs. 7-38). Nueva York: Springer .

Hayashi, T., Yamasaki, S., Morita, N., & Aida, H. (1999). Effects of IP packet loss and picture frame reduction on MPEG1 subjective quality. *Multimedia Signal Processing, 1999 IEEE 3rd Workshop on* , 515 - 520 .

He, Z., & Chen, C. W. (2002). End-to-end video quality analysis and modeling for video streaming over IP network. *Multimedia and Expo, 2002. ICME '02. Proceedings. 2002 IEEE International Conference on , 1*, 853 - 856 .

Internet Society. (2015). *Global Internet Report 2015.*

Kharagpur, I. (13 de Diembre de 2009). *Broadcast Communication Networks* . Obtenido de National Programme on Technology Enhanced Learning : http://www.nptel.ac.in/courses/106105080/pdf/M5L1.pdf

Liu, Z., Qiao, Y., Karunakar, A. K., Lee, B., Fallon, E., Zhang, C., y otros. (Septiembre de 2015). H.264/MVC interleaving for real-time multiview video streaming. *Journal of Real-Time Image Processing, 10*(3), 501-511.

Mack, S., & Rayburn, D. (2005). *Hands-On Guide to Webcasting: Internet Event and AV Production.* Estados Unidos: Focal Press.

Maly, R. J. (2003). *Comparison of Centralized (Client-Server) and Decentralized (Peer-to-Peer) Networking.* Tesis semestral, Swiss Federal Institute of Technology, Computer Engineering and Networks Laboratory, Zurich, Switzerland.

Mansour, Y., & Patt-Shamir, B. (Agosto de 2001). Jitter Control in QoS Networks. *IEEE/ACM Transactions on Networking , 9*(4), 492-502.

Marsic, I. (11 de Junio de 2013). *Computer Networks: Performance and Quality of Service.* Obtenido de Rutgers: School of Engineering: http://www.ece.rutgers.edu/~marsic/books/CN/book-CN_marsic.pdf

McMillan, T. (2015). *Cisco Networking Essentials, 2nd Edition.* Wiley & Sons Ltd.

Meador, B. (24 de Noviembre de 2008). *A Survey of Computer Network Topology and*. Obtenido de Washington University in St. Louis School of Engineering & Applied Science, Department of Computer Science & Engineering: http://www.cse.wustl.edu/~jain/cse567-08/ftp/topology.pdf

Meinel, C., & Sack, H. (2013). *Internetworking: Technological Foundations and Applications.* Springer.

Mohan, V., Reddy, Y. R., & Kalpana, K. (Agosto de 2011). Active and Passive Network Measurements : A Survey. *International Journal of Computer Science and Information Technologies, 2*(4), 1372-1385.

Muraoka, S., Masuyama, H., Kasahara, S., & Takahashi, Y. (Junio de 2009). FEC recovery performance for video streaming services over wired-wireless networks. *Performance Evaluation, 66*(6), 327–342.

Mwela, J. S., & Adebomi, O. E. (2010). *Impact of Packet Loss on the Quality of Video Stream Transmission .* Tesis para grado de maestría, Blekinge Institute of Technology, Computo, Suecia.

Norman, J. (2004-2016). *Jeremy Norman's History of Information*. Obtenido de http://www.historyofinformation.com/expanded.php?id=3821

Ouellet, E., Padjen, R., Pfund, A., Fuller, R., & Blankenship, T. (2002). *Building a Cisco Wireless LAN.* Estados Unidos: Syngress .

Ozer, J. (26 de Febrero de 2011). *What is Streaming?* Obtenido de StreamingMedia.com, an Information Today, Inc. (ITI) company: http://www.streamingmedia.com/Articles/Editorial/What-Is-.../What-is-Streaming-74052.aspx

Pau, T., & Ogunfunmi, T. (3 de Marzo de 2008). Wireless LAN Comes of Age: Understanding the IEEE 802.11n Amendment. *IEEE Circuits and Systems Magazine , 8*, 28-54.

Paul Baran and the Origins of the Internet. (s.f.). Obtenido de The RAND Corporation: http://www.rand.org/about/history/baran.list.html

Perkins, C., Hodson, O., & Hardman, V. (Sept.-Oct. de 1998). A Surver of Packet Loss Recovery Techniques for Streaming Audio. *IEEE Network, 12*(5), 40-48.

Poellabauer, C. (Septiembre de 2014). *Introducing Basic Network Concepts.* Obtenido de University of Notre Dame: http://www3.nd.edu/~cpoellab/teaching/cse40814_fall14/networks.pdf

Ponlatha, S., & Sabeenian, R. S. (Diciembre de 2013). Comparison of Video Compression Standards. *International Journal of Electrical and Computer Engineering, 5*(6), 549-554.

QUERCIA, D. (2002). *A SIMULATIVE STUDY OF DISTRIBUTED SPEECH RECOGNITION OVER INTERNET PROTOCOL NETWORKS.* Tesis de maestría, Universidad de Illinois , Ingeniería Eléctrica y Computación, Chicago.

Raake, A. (2006). *Speech Quality of VoIP: assessment and prediction.* John Wiley & Sons Ltd.

Richardson, I. E. (Junio de 1995). Usage parameter control cell loss effects on MPEG video. *Communications, 1995. ICC '95 Seattle, 'Gateway to Globalization', 1995 IEEE International Conference on, 2*, 970 - 974.

Richardson, I. E. (2003). *H.264 and MPEG-4 Video Compression.* Wiltshire, Inglaterra: John Wiley & Sons Ltd.

Rigotti, G. (s.f.). *El modelo OSI.* Obtenido de Facultad de Ciencias Exactas – UNICEN: http://www.exa.unicen.edu.ar/catedras/comdat1/material/ElmodeloOSI.pdf

Schulzrinne, H., Casgner, S., Frederick, R., & Jacobson, V. (Julio de 2003). *RTP: A Transport Protocol for Real-Time Applications*. Obtenido de Request for Comments: 3550: https://tools.ietf.org/html/rfc3550

Schulzrinne, H., Rao, A., & Lanphier, R. (Abril de 1998). *Real Time Streaming Protocol* . Obtenido de Request for Comments: 2326: https://tools.ietf.org/html/rfc2326

Sikora, T. (1997). MPEG Digital Audio-and Video-Coding Standards. *IEEE Signal Processing Magazine, 14*(5), 82-100.

Simoneau, P. (2006). *The OSI Model: Understanding the Seven Layers of Computer Networks.* Obtenido de Global Knowledge: http://faculty.spokanefalls.edu/Rudlock/files/WP_Simoneau_OSIModel.pdf

Soldani, D., Li, M., & Cuny, R. (2006). *QoS and QoE Management in UMTS Cellular Systems* (Primera ed.). Londres, Reino Unido: John Wiley & Sons, Ltd.

Szigeti, T., Hattingh, C., & Barton, R. (2013). *End-to-End QoS Network Design: Quality of Service for Rich-Media & Cloud Networks (Networking Technology)* (Segunda ed.). Cisco Press.

T. Friedman, E., R. Caceres, E., & A. Clark, E. (Noviembre de 2003). *RTP Control Protocol Extended Reports (RTCP XR)*. Recuperado el 12 de Mayo de 2016, de https://tools.ietf.org/html/rfc3611

Telchemy. (s.f.). *Application Performance Management*. Obtenido de Telchemy: https://www.telchemy.com/index.php

The R Foundation. (s.f.). *What is R?* Obtenido de The R Foundation: https://www.r-project.org/about.html

UIT-T. (Marzo de 1993). *H.261 : Códec vídeo para servicios audiovisuales a p x 64 kbit/s.* Obtenido de Unión Internacional de Telecomunicaciones: https://www.itu.int/rec/T-REC-H.261-199303-I/es

VideoLAN Organization. (s.f.). *VLC media player* . Obtenido de VideoLAN Organization: https://www.videolan.org/vlc/

Wetherall, D. J., & S.Tanenbaum, A. (2011). *Computer Networks* (Quinta ed.). Pearson.

Wireshark Foundation . (s.f.). *About Wireshark*. Obtenido de Wireshark: https://www.wireshark.org/

Wu, D., Hou, Y. T., & Zhang, Y.-Q. (Diciembre de 2000). Transporting real-time video over the Internet: challenges and approaches. *Proceedings of the IEEE, 88*(12), 1855 - 1877.

Wu, D., Hou, Y. T., Zhu, W., Shanz, Y.-Q., & Peha, J. (Marzo de 2001). Streaming video over the Internet: approaches and directions. *IEEE Transactions on Circuits and Systems for Video Technology, 11*(3), 282 - 300.

Zambelli, A. (1 de Marzo de 2013). *A history of media streaming and the future of connected TV* . Obtenido de Guardian News and Media Limited: http://www.theguardian.com/media-network/media-network-blog/2013/mar/01/history-streaming-future-connected-tv

Zhu, W., Hou, Y. T., Wang, Y., & Zhang, Y.-Q. (Junio de 1997). Modeling and simulation of MPEG-2 video transport over ATM networks considering the jitter effect . *Multimedia Signal Processing, 1997., IEEE First Workshop on* , 401 - 406 .

Apéndice

ANEXO A.1

```
@ECHO off
Setlocal EnableDelayedExpansion
REM  Defina el directorio donde estan almacenados los archivos .pcap
set dir=D:\capturasFiltradas\Mediciones\Cliente\MPEG1\MPEG1_40mhz_Dia_1_FrecuenciaAuto\


set dir2= %dir%Parametros\
if not exist "%dir%Parametros" mkdir %dir2%

REM  Ingrese la direccion de localizacion de wireshark
cd C:\Program Files\Wireshark\

REM  Con este ciclo nos movemos entre los archivos .pcap, modificarlo si se requiere.
REM  Para este trabajo se recolecto 12 archivos .pcap, por lo tanto el ciclo va de 1,12

               for /l %%x in (1, 1, 12) do (
        echo "%dir%%%x.pcap"
        echo "Archivo de video: %%x.pcap"

        REM  comandos propios de tshark, acceda a la documentacion en linea para saber más
        REM  rtp.p type=32 hace referencia al flujo de video del codec video MPEG, este numero puede cambiar
dependiendo del codec de audio o video utilizado
        REM  Y "rtsp && sdp" es el filtro para identificar el paquete del protocolo SDP (Session Description
Protocol), se requiere este paquete para obtener el sdp.sample_rate que se utilizara para calcular el jitter
        REM  los campos bajo el comando -e son los parametros basicos de cada paquete usados para calcular jitter y
perdidas
         tshark -Y "rtsp && sdp" -T fields -n -r "%dir%%%x.pcap" -E header=y -E separator=, -e frame.len -e
udp.length -e udp.srcport -e rtp.seq -e frame.time_epoch -e rtp.timestamp -e sdp.sample_rate >>
"%dir%\Parametros\Packet_%%x_v.csv"
         tshark -Y "rtp.p_type == 32" -T fields -n -r "%dir%%%x.pcap" -E header=n -E separator=, -e frame.len -e
udp.length -e udp.srcport -e rtp.seq -e frame.time_epoch -e rtp.timestamp -e sdp.sample_rate >>
"%dir%Parametros\Packet_%%x_v.csv"

               echo "%dir%%%x.pcap"
               echo "Archivo de audio: %%x.pcap"

        REM  comandos propios de tshark, acceda a la documentacion en linea para saber más
        REM  rtp.p type=14 hace referencia al flujo de audio del codec video MPEG, este numero puede cambiar
dependiendo del codec de audio o video utilizado
        REM  Y "rtsp && sdp" es el filtro para identificar el paquete del protocolo SDP (Session Description
Protocol), se requiere este paquete para obtener el sdp.sample_rate que se utilizara para calcular el jitter
        REM  los campos bajo el comando -e son los parametros basicos de cada paquete usados para calcular jitter y
perdidas
        tshark -Y "rtsp && sdp" -T fields -n -r "%dir%%%x.pcap" -E header=y -E separator=, -e frame.len -e
udp.length -e udp.srcport -e rtp.seq -e frame.time_epoch -e rtp.timestamp -e sdp.sample_rate >>
"%dir%Parametros\Packet_%%x_a.csv"
        tshark -Y "rtp.p_type == 14" -T fields -n -r "%dir%%%x.pcap" -E header=n -E separator=, -e frame.len -e
udp.length -e udp.srcport -e rtp.seq -e frame.time_epoch -e rtp.timestamp -e sdp.sample_rate >>
"%dir%Parametros\Packet_%%x_a.csv"

)

pause
exit
```

Script 1 Extracción de datos en Wireshark

ANEXO A.2

```
############Scrip para calcular Jitter ###################
#### Este script calcula el Jitter de un archivo .pcap que contenga flujos de video streaming (Video/Audio)
#### Si solo desea calcular el jitter para un tipo especifico de flujo, ya sea solo video o solo audio, proceda a
#### eliminar la seccion que no desea calcular (audio o video) podra identificarlos con las terminaciones ".v" y
".a"
#### Es necesario saber el clock rate del codec que se utilizo. Es posible obtenerlo con el paquete de Session
Description Protocol (SDP)
#### recibido en el archivo .pcap o accediendo a la información de RFC del codec
######## Elaborado por Christian Vadillo


###########                                          ###########
########### ASIGNE LOS SIGUIENTES VALORES ###########
###########                                          ###########

#### Asignar el directorio de trabajo en donde están almacenadas los .csv
dir<-'D:/capturasFiltradas/Mediciones/Cliente/MPEG1/MPEG1_5mhz_Dia_4_FrecuenciaAuto/Parametros/'
###########                                          ###########
newfilepath <- file.path(dir)
setwd(newfilepath)

#### Defina cuántos archivos se analizaran por tipo, por ejemplo,
#### Para este trabajo se midio 12 horas por día, por lo que hay 12 archivos
#### por flujo de video y 12 archivos para flujo de audio.
#### De tal forma definimos Num_Archivos con valor 12

Num_Archivos<-12

###########           ###########                  ###########
###########           ###########                  ###########
###########           ###########                  ###########


#### Leer los .csv
archivos = list.files(path=newfilepath,pattern="*.csv")


#### Importando los .csv a R
for (i in 1:length(archivos ))
assign(paste("Parametros_",archivos [i],sep = ""), read.csv(archivos[i],skip=1, header = FALSE,na.strings =
"NA",col.names=c("frame.len","udp.length","udp.srcport","rtp.seq","frame.time_epoch","rtp.timestamp","sdp.sample_r
ate.v","sdp.sample_rate.a")))

Parametros.v <- list()
Parametros.a <- list()

#### Mover los .csv a las listas  ###Verificar el nombre de los .csv
for (i in 1:Num_Archivos)
{
archivo.v = paste("Parametros_Packet_", i,"_v.csv", sep = "")
archivo.a = paste("Parametros_Packet_", i,"_a.csv", sep = "")
Parametros.v[[i]] <- eval(parse(text=archivo.v))
Parametros.a[[i]] <- eval(parse(text=archivo.a))
}

###Asigar el RTP timestamp clock rate del codec, Por ejemplo, para este trabajo se utilizo MPEG-1 cuyo clock rate
es 90000 kHz
rtp.clock.v <- 1/Parametros.v[[1]]$sdp.sample_rate.v[1]   #Frecuencia del codec de video
rtp.clock.a <- 1/Parametros.a[[1]]$sdp.sample_rate.a[1]     #Frecuencia del codec de audio

###En caso de que quiera asignarlo manualmente, comente la parte de arriba y active las dos lineas sig:
#rtp.clock.v <- 1/90000   #Frecuencia del codec de video
#rtp.clock.a <- 1/90000       #Frecuencia del codec de audio



#### Creación de listas para almacenar jitter
Jitter.v<-list()
Jitter.a<-list()
```

Script 2. 1 Jitter.r

```
############################## PASO 2. CALCULAR JITTER VIDEO ############################

################ Ciclo para movernos entre los archivos
for(i in 1:Num_Archivos)
{
        jitter_all<-c()               #vector que guardara momentaneamente el jitter de 1 archivo
        delta_all<-c()          #vector que guardara momentaneamente el delta de 1 archivo
        posVector = 1  ### Para movernos entre el vetor 'jitter'

        PORTS <- unique(Parametros.v[[i]]$udp.srcport)      ### Extraemos cuantos puertos diferentes hay en 1
archivo

        Rl<-0           ### Tiempo de llegada del primer paquete
        Sl<-0                   ### Tiemstamp del primer paquete
        Rll<-0          ### Tiempo de llegada del segundo paquete
        Sll<-0         ### Tiemstamp del segundo paquete

### Dado que al detectarse un nuevo flujo de video o de audio, ya sea por comandos externos del reproductor o
### por eventos importantes de perdidas de paquetes, se genera un nuevo puerto udp, por lo que, al generarse un
### nuevo puerto, los puntos de referencia para el calculo de jitter se reinician. Para evitar problemas con este
evento, se
### va a separar los paquetes por su puerto udp, con esto excluimos tener que verificar si ha habido un cambio de
puerto en
### el flujo de video o audio

        for(a in 1:length(PORTS))  ### Para moverse entre los puertos udp
                {
                data.v<- (subset(Parametros.v[[i]], udp.srcport==PORTS[a]))  ## subdivision de la base de datos
completa, donde solo estaran los paquetes del puerto en curso
                Recibidos <- length(data.v$rtp.seq)         ## Paquetes recibidos bajo el puerto udp en turno
                posDato = 1
                jitter_all[posVector]=0                             ## Por cada nuevo puerto udp detectado, se
reinician los parametros de jitter
                delta_all[posVector]=0                      ## Por cada nuevo puerto udp detectado, se reinician
los parametros de delta
                posVector=posVector+1

 ## Mientras posDato sea menor que los paquetes recibidos
                while (posDato <Recibidos)
                        {
                                Rll=(data.v$frame.time_epoch[posDato+1])*1000
                                Rl=(data.v$frame.time_epoch[posDato])*1000
                                Sl=(data.v$rtp.timestamp[posDato]*rtp.clock.v)*1000

                                        ##¿Se ha ciclado la secuencia de timestamp?
                                if (data.v$rtp.timestamp[posDato]>(data.v$rtp.timestamp[posDato+1]+1))
                                        {
                                        ##cat("\n\n Ciclo detectado:",posDato)
                                                Sll =
(((data.v$rtp.timestamp[posDato+1]+0xffffffff)*rtp.clock.v)*1000)
                                        }else{
                                                Sll=(data.v$rtp.timestamp[posDato+1]*rtp.clock.v)*1000
                                                }

                #Se calcula Delta
                                        delta_all[posVector] = (Rll-Rl)-(Sll-Sl)
                #Se calcula el Jitter en milisegundos del flujo RTP
                                        jitter_all[posVector]=(jitter_all[posVector-1]+((abs(delta_all[posVector])-
jitter_all[posVector-1])/16))
                                        posDato=posDato+1
                                        posVector=posVector+1
                        }
cat("\n\nPort finalizado:", PORTS[a])

                }
### movemos los valores calculados de 1 archivo a la lista Jitter.v cuya posicion sera 'i', o bien, el numero de
archivo analizado
Jitter.v[[i]] <- data.frame(ID=c(1:length(jitter_all)),Delta=c(delta_all),Jitter=c(jitter_all))
cat ("Archivos analizados para flujo de video: ", i,"\n")
}

head(Jitter.v[[i]]) ### para observar los primeros datos de jitter calculados
tail(Jitter.v[[i]]) ### para observar los ultimos datos de jitter calculados

#SEPARA LA LISTA EN TABLAS
for (i in seq(Jitter.v))
  assign(paste("Jitter_V_", i, sep = ""), Jitter.v[[i]])
```

Script 2. 2 Jitter.r

```
############################## PASO 2. CALCULAR JITTER AUDIO ############################
################ Ciclo para movernos entre las horas ###################################################################
for(i in 1:Num_Archivos)
{
        jitter_all<-c()              #vector que guardara momentaneamente el jitter de 1 archivo
        delta_all<-c()         #vector que guardara momentaneamente el delta de 1 archivo
        posVector = 1  ### Para movernos entre el vetor 'jitter'

        PORTS <- unique(Parametros.a[[i]]$udp.srcport) ### Extraemos cuantos puertos diferentes hay en 1 archivo

        Rl<-0           ### Tiempo de llegada del primer paquete
        Sl<-0                ### Tiemstamp del primer paquete
        Rll<-0          ### Tiempo de llegada del segundo paquete
        Sll<-0         ### Tiemstamp del segundo paquete

### Dado que al detectarse un nuevo flujo de video o de audio, ya sea por comandos externos del reproductor o
### por eventos importantes de perdidas de paquetes, se genera un nuevo puerto udp, por lo que, al generarse un
### nuevo puerto, los puntos de referencia para el calculo de jitter se reinician. Para evitar problemas con este
evento, se
### va a separar los paquetes por su puerto udp, con esto excluimos tener que verificar si ha habido un cambio de
puerto en
### el flujo de video o audio

        for(a in 1:length(PORTS))  ### Para moverse entre los puertos udp
                {
                data.a<- (subset(Parametros.a[[i]], udp.srcport==PORTS[a])) ## subdivision de la base de datos
completa, donde solo estaran los paquetes del puerto en curso
                Recibidos <- length(data.a$rtp.seq)          ## Paquetes recibidos bajo el puerto udp en turno
                posDato = 1
                jitter_all[posVector]=0                              ## Por cada nuevo puerto udp detectado, se
reinician los parametros de jitter
                delta_all[posVector]=0                       ## Por cada nuevo puerto udp detectado, se reinician
los parametros de delta
                posVector=posVector+1

  ## Mientras posDato sea menor que los paquetes recibidos
                while (posDato <Recibidos)
                        {

                                Rll=(data.a$frame.time_epoch[posDato+1])*1000
                                Rl=(data.a$frame.time_epoch[posDato])*1000
                                Sl=(data.a$rtp.timestamp[posDato]*rtp.clock.a)*1000

                                        ##¿Se ha ciclado la secuencia de timestamp?
                                if (data.a$rtp.timestamp[posDato]>(data.a$rtp.timestamp[posDato+1]+1))
                                        {
                                        ##cat("\n\n Ciclo detectado:",posDato)
                                                Sll =
(((data.a$rtp.timestamp[posDato+1]+0xffffffff)*rtp.clock.a)*1000)
                                        }else{
                                                Sll=(data.a$rtp.timestamp[posDato+1]*rtp.clock.a)*1000
                                                }

                #Se calcula Delta
                                delta_all[posVector] = (Rll-Rl)-(Sll-Sl)
                #Se calcula el Jitter en milisegundos del flujo RTP
                                jitter_all[posVector]=(jitter_all[posVector-1]+((abs(delta_all[posVector])-
jitter_all[posVector-1])/16))
                                posDato=posDato+1
                                posVector=posVector+1
                        }
cat("\n\nPort finalizado:", PORTS[a])

                }
### movemos los valores calculados de 1 archivo a la lista Jitter.a cuya posicion sera 'i', o bien, el numero de
archivo analizado
Jitter.a[[i]] <- data.frame(ID=c(1:length(jitter_all)),Delta=c(delta_all),Jitter=c(jitter_all))
cat ("Archivos analizados para flujo de audio: ", i,"\n")
}

head(Jitter.a[[i]])
tail(Jitter.a[[i]])

#SEPARA LA LISTA EN TABLAS
for (i in seq(Jitter.a))
  assign(paste("Jitter_A_", i, sep = ""), Jitter.a[[i]])
```

Script 2. 3 Jitter.r

```
########### PASO 4. SACAR MEDIAS POR HORA JITTER - VIDEO ##############
#####################################################################
Promedio_Jitter_V<- data.frame(Hora=1:Num_Archivos,Jitter_Promedio=1:Num_Archivos)
Promedio_Jitter_A<- data.frame(Hora=1:Num_Archivos,Jitter_Promedio=1:Num_Archivos)

        for (i in 1:Num_Archivos)
        {
            # Promedio video
                Promedio_Jitter_V$Jitter_Promedio[i] = mean(Jitter.v[[i]]$Jitter)
            #Promedio Audio
                Promedio_Jitter_A$Jitter_Promedio[i] = mean(Jitter.a[[i]]$Jitter)
        }


########### PASO 5. CREAR ARCHIVOS .CSV ##############
path_user <- dir
subDir <- "Jitter"
dir.create(file.path(path_user, subDir), showWarnings = FALSE)


newfilename <- "Promedio_Jitter_V.csv"
newfilepath <- file.path(path_user,subDir, newfilename)
write.table(Promedio_Jitter_V, newfilepath ,sep = ",", row.names = FALSE)


newfilename <- "Promedio_Jitter_A.csv"
newfilepath <- file.path(path_user,subDir, newfilename)
write.table(Promedio_Jitter_A, newfilepath ,sep = ",", row.names = FALSE)


        for (i in 1:Num_Archivos)
        {
newfilename  <-paste("Jitter_V_Hora_", i,".csv", sep = "")
newfilepath <- file.path(path_user,subDir, newfilename)
write.table(Jitter.v[[i]]$Jitter, newfilepath ,sep = ",", row.names = FALSE)


newfilename  <-paste("Jitter_A_Hora_", i,".csv", sep = "")
newfilepath <- file.path(path_user,subDir, newfilename)
write.table(Jitter.a[[i]]$Jitter, newfilepath ,sep = ",", row.names = FALSE)
        }
```

Script 2. 4 Jitter.r

ANEXO A.3

```
############Script para calcular perdidas ###################
######## Elaborado por Christian Vadillo

#### Asignar el directorio de trabajo en donde están almacenadas los .csv
dir<-'D:/capturasFiltradas/Mediciones/Cliente/MPEG1/MPEG1_10mhz_Dia_3_FrecuenciaAuto/Parametros/'
newfilepath <- file.path(dir)
setwd(newfilepath)

########### CREAR ARCHIVOS .CSV ##############

#### Defina cuántos archivos se analizaran por tipo, por ejemplo,
#### Para este trabajo se midio 12 horas por día, por lo que hay 12 archivos
#### por flujo de video y 12 archivos para flujo de audio.
#### De tal forma definimos Num_Archivos con valor 12

Num_Archivos<-12


#### Leer los .csv
newfilepath <- file.path(dir)
archivos = list.files(path=newfilepath,pattern="*.csv")

#### Importando los .csv a R
for (i in 1:length(archivos ))
assign(paste("Parametros_",archivos [i],sep = ""), read.csv(archivos[i],skip=1, header = FALSE,na.strings =
"NA",col.names=c("frame.len","udp.length","udp.srcport","rtp.seq","frame.time_epoch","rtp.timestamp","sdp.sample_rat
e.v","sdp.sample_rate.a")))

Parametros.v <- list()
Parametros.a <- list()


#### Mover los .csv a las listas  ###Verificar el nombre de los .csv
for (i in 1:Num_Archivos)
{
archivo.v = paste("Parametros_Packet_", i,"_v.csv", sep = "")
archivo.a = paste("Parametros_Packet_", i,"_a.csv", sep = "")
Parametros.v[[i]] <- eval(parse(text=archivo.v))
Parametros.a[[i]] <- eval(parse(text=archivo.a))
}

#### Creación de listas

Perdidas.v<-list()
Perdidas.a<-list()
```

Script 3. 1 Perdidas.r

```
############################## PASO 2. Obtener paquetes perdidos - VIDEO
for(i in 1:Num_Archivos)  ### Para movernos entre las archivos
{
### Dado que al detectarse un nuevo flujo de video o de audio, ya sea por comandos externos del reproductor o
### por eventos importantes de pérdidas de paquetes, se genera un nuevo puerto udp, por lo que, al generarse un
### nuevo puerto, los puntos de referencia cambian. Para evitar problemas con este evento, se
### va a separar los paquetes por su puerto udp, con esto excluimos tener que verificar si ha habido un cambio de
puerto en el flujo de video o audio
PORTS <- unique(Parametros.v[[i]]$udp.srcport)
        paquetes=c()      ### Almacenará los 1 y 0 según corresponda, (1=recibido, 0=perdido)
        cant_perdidos<-0  ### Sí se presenta una pérdida, esta variable almacenará cuántos paquetes perdidos hubo
        posVector = 1  ### Para movernos entre el vetor 'paquetes'
PER=  0               ### Para mostrar las perdidas en la linea de comandos
for(a in 1:length(PORTS)) ## Para moverse entre los puertos udp
{
        data.v<- (subset(Parametros.v[[i]], udp.srcport==PORTS[a]))  ## subdivisión de la base de datos completa,
donde solo estarán los paquetes del puerto en curso
        Recibidos <- length(data.v$rtp.seq)
        posDato = 1           ### Para movernos entre la base de datos de los paquetes
              ###Mientras la posición no supere los paquetes recibidos
        while (posDato <Recibidos)
        {                     ### Condición para determinar que el paquete fue recibido.
              if (data.v$rtp.seq[posDato+1]- data.v$rtp.seq[posDato]==1)
              {
                      paquetes[posVector]<-1
                      posDato=posDato+1
                      posVector=posVector+1
              }
               else { ### Condición para determinar si se ha ciclado las secuencias (rango de ciclo= 0 a 65535).
                              if ((data.v$rtp.seq[posDato]==65535 && data.v$rtp.seq[posDato+1]==0) ||
(data.v$udp.srcport[posDato+1]==data.v$udp.srcport[posDato] && data.v$rtp.seq[posDato+1]-data.v$rtp.seq[posDato]<0))
                                      {
                                      paquetes[posVector]<-1
                                      posDato=posDato+1
                                      posVector=posVector+1
                                      }
                                      else{
                                              ### Condición para determinar si hay paquetes duplicados.
                                              if ((data.v$rtp.seq[posDato]==data.v$rtp.seq[posDato+1]))
                                                      {
                                                              paquetes[posVector]<-1
                                                              posDato=posDato+1
                                                              posVector=posVector+1

                                                      }
                                                      else{
              ### Si no se cumplen ninguna de las condiciones previas, se entiende que ha habido una perdida.
                                                      cant_perdidos= (data.v$rtp.seq[posDato+1]-
data.v$rtp.seq[posDato])-1
                                                              #cat("\n\nSe perdieron paquetes  = ", cant_perdidos,
"\nPosición=",posDato,"\n")
                                                              paquetes[posVector]<-1
                                                              posVector=posVector+1
                                                              for (cont in 1:cant_perdidos)
                                                                      {
                                                                      paquetes[posVector]<-0
                                                                      posVector=posVector+1
                                                                      }
                                                              posDato=posDato+1
                                                              }
                                              }
                              }

        }
cat("\n\nPort finalizado:", PORTS[a])
cat("\nPerdidos: ", sum(paquetes==0) - PER)
PER=  sum(paquetes==0)
}
cat("Length Sequence = ", length(Parametros.v[[i]]$rtp.seq) , "\n")
cat("Length paquetes = ", length(paquetes) , "\n")
cat("hora = ", i , "\n")
cat("posDato = ", posDato , "\n")
cat("posVector = ", posVector , "\n")
cat("TotalPerdidos = ", length(paquetes)- length(Parametros.v[[i]]$rtp.seq) , "\n")
Perdidas.v[[i]] <- data.frame(ID=c(1:length(paquetes)),Paquetes=c(paquetes))
library(Rcmdr)
#### Clasificar los paquetes en Recibido y Perdido
Perdidas.v[[i]]$"Estado" <-Recode(Perdidas.v[[i]]$Paquetes,'1="Recibido";0="Perdido"', as.factor.result=TRUE)
}
```

Script 3. 2 Perdidas.r

```
############################## PASO 2. Obtener paquetes perdidos - AUDIO
for(i in 1:Num_Archivos)  ### Para movernos entre las horas
{
PORTS <- unique(Parametros.a[[i]]$udp.srcport)
       paquetes=c()       ### Almacenará los 1 y 0 según corresponda, (1=recibido, 0=perdido)
       #paquetes[1]<-1        ### El primer paquete siempre será recibido
       cant_perdidos<-0  ### Sí se presenta una pérdida, esta variable almacenará cuántos paquetes perdidos hubo
       posVector = 1  ### Para movernos entre el vetor 'paquetes'
PER=  0
for(a in 1:length(PORTS))
{
       data.a<- (subset(Parametros.a[[i]], udp.srcport==PORTS[a]))
       Recibidos <- length(data.a$rtp.seq)
       posDato = 1            ### Para movernos entre la base de datos de los paquetes
              ###Mientras la posición no supere los paquetes recibidos
       while (posDato <Recibidos)
       {                      ### Condición para determinar que el paquete fue recibido.
### Primero verificamos que se siga en el mismo flujo de video, ya que cada flujo nuevo, genera otro src.port
              if (data.a$rtp.seq[posDato+1]- data.a$rtp.seq[posDato]==1)
              {
                     paquetes[posVector]<-1
                     posDato=posDato+1
                     posVector=posVector+1
              }
               else { ### Condición para determinar si se ha ciclado las secuencias (rango de ciclo= 0 a 65535).
                            if ((data.a$rtp.seq[posDato]==65535 && data.a$rtp.seq[posDato+1]==0) ||
(data.a$udp.srcport[posDato+1]==data.a$udp.srcport[posDato] && data.a$rtp.seq[posDato+1]-data.a$rtp.seq[posDato]<0))
                                   {
                                   paquetes[posVector]<-1
                                   posDato=posDato+1
                                   posVector=posVector+1

                                   }
                                   else{

                                          if ((data.a$rtp.seq[posDato]==data.a$rtp.seq[posDato+1]))
                                                 {
                                                        paquetes[posVector]<-1
                                                        posDato=posDato+1
                                                        posVector=posVector+1

                                                 }
                                                 else{
              ### Si no se cumplen ninguna de las condiciones previas, se entiende que ha habido una perdida .
                                                 cant_perdidos= (data.a$rtp.seq[posDato+1]-
data.a$rtp.seq[posDato])-1
                            #cat("\n\nSe perdieron paquetes  = ", cant_perdidos, "\nPosición=",posDato,"\n")
                                                        paquetes[posVector]<-1
                                                        posVector=posVector+1
                                                        for (cont in 1:cant_perdidos)
                                                               {
                                                               paquetes[posVector]<-0
                                                               posVector=posVector+1
                                                               }
                                                        posDato=posDato+1
                                                        }
                                   }
                     }

       }
cat("\n\nPort finalizado:", PORTS[a])
cat("\nPerdidos: ", sum(paquetes==0) - PER)
PER=  sum(paquetes==0)
}
cat("Length Sequence = ", length(Parametros.v[[i]]$rtp.seq) , "\n")
cat("Length paquetes = ", length(paquetes) , "\n")
cat("hora = ", i , "\n")
cat("posDato = ", posDato , "\n")
cat("posVector = ", posVector , "\n")
cat("TotalPerdidos = ", length(paquetes)- length(Parametros.a[[i]]$rtp.seq) , "\n")

Perdidas.a[[i]] <- data.frame(ID=c(1:length(paquetes)),Paquetes=c(paquetes))
library(Rcmdr)
#### Clasificar los paquetes en Recibido y Perdido
Perdidas.a[[i]]$"Estado" <-Recode(Perdidas.a[[i]]$Paquetes,'1="Recibido";0="Perdido"', as.factor.result=TRUE)
}
```

Script 3. 3 Perdidas.r

```
########### SACAR PORCENTAJES
#####################################################################
Perdidas_V<- data.frame(Hora=1:Num_Archivos,Recibidos=1:Num_Archivos,Perdidos=1:Num_Archivos,
Total_Enviados=1:Num_Archivos)
Perdidas_A<- data.frame(Hora=1:Num_Archivos,Recibidos=1:Num_Archivos,Perdidos=1:Num_Archivos,
Total_Enviados=1:Num_Archivos)

       for (i in 1:Num_Archivos)
       {
          # Total video
              Perdidas_V$Total_Enviados[i] = (sum(Perdidas.v[[i]]$Paquetes=="0") +
sum(Perdidas.v[[i]]$Paquetes=="1"))
          # Recibidos video
              Perdidas_V$Recibidos[i] = ((sum(Perdidas.v[[i]]$Paquetes=="1") * 100) / Perdidas_V$Total_Enviados[i])
          # Perdidos video
              Perdidas_V$Perdidos[i] = ((sum(Perdidas.v[[i]]$Paquetes=="0") * 100) / Perdidas_V$Total_Enviados[i])


          # Total Audio
              Perdidas_A$Total_Enviados[i] = (sum(Perdidas.a[[i]]$Paquetes=="0") +
sum(Perdidas.a[[i]]$Paquetes=="1"))
          # Recibidos Audio
              Perdidas_A$Recibidos[i] = ((sum(Perdidas.a[[i]]$Paquetes=="1") * 100) / Perdidas_A$Total_Enviados[i])
          # Perdidos Audio
              Perdidas_A$Perdidos[i] = ((sum(Perdidas.a[[i]]$Paquetes=="0") * 100) / Perdidas_A$Total_Enviados[i])

       }



########### CREAR ARCHIVOS .CSV ##############
path_user <- dir
subDir <- "Perdidas"
dir.create(file.path(path_user, subDir), showWarnings = FALSE)


newfilename <- "Perdidas_V.csv"
newfilepath <- file.path(path_user,subDir, newfilename)
write.table(Perdidas_V, newfilepath ,sep = ",", row.names = FALSE)


newfilename <- "Perdidas_A.csv"
newfilepath <- file.path(path_user,subDir, newfilename)
write.table(Perdidas_A, newfilepath ,sep = ",", row.names = FALSE)
```

Script 3. 4 Perdidas.r

ANEXO A.4

```
#### CALCULAR RÁFAGAS DE PÉRDIDAS
####Elaborado por Christian Vadillo
#### Se necesita haber calculado previamente los paquetes perdidos con el script 'Perdidas.r'
library(dplyr)

#### Listas para almacenar las ráfagas
Rafagas.v <- list()
Rafagas.a <- list()

#### Defina cuántos archivos se analizaran por tipo, por ejemplo,
#### Para este trabajo se midio 12 horas por día, por lo que hay 12 archivos
#### por flujo de video y 12 archivos para flujo de audio.
#### De tal forma definimos Num_Archivos con valor 12

Num_Archivos<-12

###################VIDEO############################

                for (i in 1:Num_Archivos) #### Para movernos entre las horas
        {
                rafaga<- 0      #### Contará los paquetes pertenecientes a una ráfaga
                posDato<-1              #### Para movernos entre la base de datos
                Recibidos <- length(Parametros.v[[i]]$rtp.seq)

        #### Por practicidad, se cambiará el valor de los paquetes perdidos de valor 0 a valor 1, y de los
        #### recibidos de valor 1 a 10, esto para aislar los paquetes perdidos y facilitar la contabilidad
                data=subset(Perdidas.v[[i]])
                data$Paquetes[data$Paquetes == 1] <- 10
                data$Paquetes[data$Paquetes == 0] <- 1

                while (posDato< Recibidos)
                {
                        if (data$Paquetes[posDato]==1)  ### Si se encuentra un paquete perdido (valor = 1)
                                {
                                posicion=posDato
                                        while (data$Paquetes[posDato]==1)  ### Cuenta los paquetes perdidos consecutivos
hasta
                                                {                                       ### que se encuentre un paquete
recibido
                                                        rafaga= rafaga+1
                                                        data$Paquetes[posDato]=0 ### Elimina los paquetes perdidos
consecutivos
                                                        posDato=posDato+1               ### a excepción del primer paquete
perdido en la ráfaga
                                                }
                                data$Paquetes[posicion]=rafaga              ### sustituye el primer paquetes pérdidos en
la ráfaga
                                rafaga<- 0                                          ### por el total de paquetes
perdidos contabilizados
                                }
                        posDato=posDato+1
                }
                data$Paquetes[data$Paquetes == 10] <- 0      ### Cambiar los paquetes recibidos a valor 0
                Rafagas.v[[i]] <- data                       ##### Agrega la nueva base de datos a la lista
Rafagas.v[[i]]
                Rafagas.v[[i]]$"Hora" <- i
                cat("Hora: ", i,"\n\n")

        }
```

Script 4. 1 Rafagas.r

```
###################AUDIO############################
                for (i in 1:Num_Archivos) #### Para movernos entre las horas
        {
                rafaga<- 0      #### Contará los paquetes pertenecientes a una ráfaga
                posDato<-1              #### Para movernos entre la base de datos
                Recibidos <- length(Parametros.a[[i]]$rtp.seq)


        #### Por practicidad, se cambiará el valor de los paquetes perdidos de valor 0 a valor 1, y de los
        #### recibidos de valor 1 a 10, esto para aislar los paquetes perdidos y facilitar la contabilidad
                data=subset(Perdidas.a[[i]])
                data$Paquetes[data$Paquetes == 1] <- 10
                data$Paquetes[data$Paquetes == 0] <- 1

                while (posDato< Recibidos)
                {
                        if (data$Paquetes[posDato]==1)  ### Si se encuentra un paquete perdido (valor = 1)
                                {
                                posicion=posDato
                                        while (data$Paquetes[posDato]==1)  ### Cuenta los paquetes perdidos consecutivos
hasta
                                                {                                       ### que se encuentre un paquete
recibido
                                                        rafaga= rafaga+1
                                                        data$Paquetes[posDato]=0 ### Elimina los paquetes perdidos
consecutivos
                                                        posDato=posDato+1               ### a excepción del primer paquete
perdido en la ráfaga
                                                }
                                data$Paquetes[posicion]=rafaga              ### sustituye el primer paquetes pérdidos en
la ráfaga
                                rafaga<- 0                                          ### por el total de paquetes
perdidos contabilizados
                                }
                        posDato=posDato+1
                }
                data$Paquetes[data$Paquetes == 10] <- 0      ### Cambiar los paquetes recibidos a valor 0
                Rafagas.a[[i]] <- data                       ##### Agrega la nueva base de datos a la lista
Rafagas.v[[i]]
                Rafagas.a[[i]]$"Hora" <- i
                cat("Hora: ", i,"\n\n")

        }
```

Script 4. 2 Rafagas.r

```
for (i in 1:Num_Archivos)
       {

####### Eliminar los paquetes recibidos de la base da datos - VIDEO
data.v=subset(Rafagas.v[[i]])
data.v[data.v == 0] <- NA
data2.v=na.omit(data.v[order(data.v$Paquetes, decreasing=FALSE),])

####### Obtener la frecuencia de las ráfagas de pérdidas
data2.v= data2.v%>% group_by(Paquetes) %>%
  summarise(count=n()) %>%
  mutate(porcentaje=count/sum(count))
  data2.v$"ID"<-1:length(data2.v$Paquetes)
  colnames(data2.v) <- c("Tamaño", "Frecuencia", "Porcentaje","ID")


####### Eliminar los paquetes recibidos de la base da datos - AUDIO
data.a=subset(Rafagas.a[[i]])
data.a[data.a == 0] <- NA
data2.a=na.omit(data.a[order(data.a$Paquetes, decreasing=FALSE),])

####### Obtener la frecuencia de las ráfagas de pérdidas
data2.a= data2.a%>% group_by(Paquetes) %>%
  summarise(count=n()) %>%
  mutate(porcentaje=count/sum(count))
  data2.a$"ID"<-1:length(data2.a$Paquetes)
  colnames(data2.a) <- c("Tamaño", "Frecuencia", "Porcentaje","ID")

#### Crear .csv#############
path_user <- dir
subDir <- "Rafagas"
dir.create(file.path(path_user, subDir), showWarnings = FALSE)

newfilename  <-paste("Rafagas_V_Hora_", i,".csv", sep = "")
newfilepath <- file.path(path_user,subDir, newfilename)
write.table(data2.v, newfilepath ,sep = ",", row.names = FALSE)

newfilename  <-paste("Rafagas_A_Hora_", i,".csv", sep = "")
newfilepath <- file.path(path_user,subDir, newfilename)
write.table(data2.a, newfilepath ,sep = ",", row.names = FALSE)
}
```

Script 4. 3 Rafagas.r