INSTITUTO POLITÉCNICO NACIONAL

UNIDAD PROFESIONAL INTERDISCIPLINARIA EN INGENIERÍA Y TECNOLOGÍAS AVANZADAS

Proyecto Terminal

"Modelado matemático, basado en cadenas de Markov, para servicios de video en vivo soportados por redes híbridas P2P-CDN"

Que para obtener el título de

"Ingeniero en Telemática"

P r e s e n t a:

Ulises Muñoz Ruiz

José Manuel Ortiz Islas

Asesores:

Dr. Mario Eduardo Rivero Ángeles

Dr. Noé Torres Cruz

Dra. Iclia Villordo Jimenez

Ciudad de México. Junio, 2023

INSTITUTO POLITÉCNICO NACIONAL

UNIDAD PROFESIONAL INTERDISCIPLINARIA EN INGENIERÍA Y TECNOLOGÍAS AVANZADAS

Proyecto Terminal

"Modelado matemático, basado en cadenas de Markov, para servicios de video en vivo soportados por redes híbridas P2P-CDN"

Que para obtener el título de:

"Ingeniero en Telemática"

P r e s e n t a:
Ulises Muñoz Ruiz
José Manuel Ortiz Islas

Asesores:

Dr. Mario Eduardo Rivero Ángeles	Dr. Noé Torres Cruz	Dra. Iclia Villordo Jimenez
Presidente del Jurado:		Secretario del Jurado:
Dr. Víctor Barrera Figueroa		Dra. Iclia Villordo Jimenez

Resumen

Live streaming es la tecnología que permite difundir una señal de vídeo a través de Internet en tiempo real. Al utilizar esta tecnología, el usuario puede visualizar videos de acuerdo a sus preferencias. Sin embargo, en esta aplicación es difícil determinar la cantidad de recursos que la red debe asignar a cada usuario para mantener una Calidad de Servicio (*QoS*, por sus siglas en inglés) adecuada. Este problema es aún más complejo cuando se consideran implementaciones en las que los usuarios funcionan como *pares* (comparten entre ellos contenido previamente descargado) y a la vez tienen acceso al video mediante Redes de Distribución de Contenido (*CDN*, por sus siglas en inglés). Considerando lo anterior, en este proyecto se desarrolló un modelo matemático, basado en una Cadena de *Markov*, que describa el funcionamiento de la red y, por lo tanto, permite evaluar su desempeño en términos de parámetros de *QoS*, así como el tiempo de descarga del video y la probabilidad de pausa en la reproducción. Además, este modelo se utiliza para evaluar parámetros de operación de la red, como el ancho de banda mínimo que debe proveer la *CDN* para garantizar la satisfacción de la *QoS*. Con base en los resultados obtenidos, también se propone un nuevo esquema de asignación de recursos, cuya eficiencia supere a esquemas actuales, en términos de los parámetros antes mencionados. Este proyecto no contempla la implementación del esquema propuesto, pero sí su evaluación con el modelo desarrollado y considerando parámetros reales de operación de las redes actuales.

Palabras Clave: Cadenas de *Markov*, Red Híbrida *CDN-P2P*, Transmisión en Vivo, *QoE*.

Índice General

Índice de Figuras

Índice de Tablas

Capítulo 1

Introducción

1.1 Introducción

En la actualidad existen dos formas de transmitir video mediante protocolos de Internet, *OTT* (*over the top*) e *IPTV* (*Internet Protocol Television*) [1]. Ambas tecnologías permiten distribuir contenido de video sobre la red, sin embargo, cada una de ellas lo realiza de manera distinta.

OTT tiene 2 enfoques: *VoD* (*Video on Demand*) y *live streaming*. En ambos casos se hace uso de servidores de Internet público para transmitir el contenido a todos los espectadores. Debido a que la calidad en la transmisión del video está sujeta a las condiciones normales de la red de internet del cliente, el archivo de video se divide en pequeños fragmentos con la finalidad de que el usuario los pueda recibir de manera cronológica, formar un *buffer* considerable y así poder reproducir el video sin complicaciones.

En *OTT* se habla de un servicio de 3 capas que garantizan una óptima experiencia de visualización de video sobre una red impredecible; la primera capa es la del video central, la segunda es la capa contenedora encargada de dividir el video en fragmentos y finalmente se presenta la capa de protocolo de transferencia basado en tiempo [1].

Figura 1. Capas de OTT

Por otro lado, *IPTV* está enfocado en hacer llegar el contenido disponible en Internet al sistema clásico de televisión, en cierto sentido se puede definir como

un sistema híbrido de *OTT* y televisión por cable. Este tipo de transmisión emplea una red *CDN* dedicada para garantizar una buena calidad de servicio, donde el video no viaja directamente a un navegador, sino que va desde un ruteador a un decodificador que permite su visualización en una TV. En este tipo de transmisión el proveedor de servicios es el responsable de proporcionar una *CDN* estable para no recurrir al servicio de proveedores terceros.

Una vez conocidas estas 2 formas de transmisión de video, el presente proyecto se enfoca en el servicio *OTT* dirigido a *live streaming*, con el propósito de desarrollar un modelo matemático que describa este tipo de sistemas y, por lo tanto, ayude a entender los fenómenos que afectan la calidad en el servicio. Además de lo anterior, también se plantea, como parte de este proyecto, definir un esquema de asignación de recursos que mejore el desempeño de propuestas publicadas con anterioridad.

La difusión de video en vivo, mejor conocido como *live streaming,* tiene como función principal permitir el acceso remoto a los usuarios que deseen conectarse a la red para acceder al contenido en tiempo real. Esto permite a los usuarios visualizar diversos eventos masivos a distancia (por ejemplo, conciertos, premiaciones, etc.), cuando su ubicación geográfica no les permite estar presentes físicamente en dicho evento. Con el ritmo tecnológico actual, este servicio debe ser flexible y adaptativo frente a las nuevas generaciones de dispositivos y el creciente número de usuarios que desean ver contenido en vivo. En el tercer trimestre del año 2019 se observó un incremento del 91.8% en el uso de las plataformas de *live streaming*. En el gráfico de la Figura 2 se puede observar el aumento en miles de personas referente al uso de la plataforma de twich en la categoría de música en vivo.

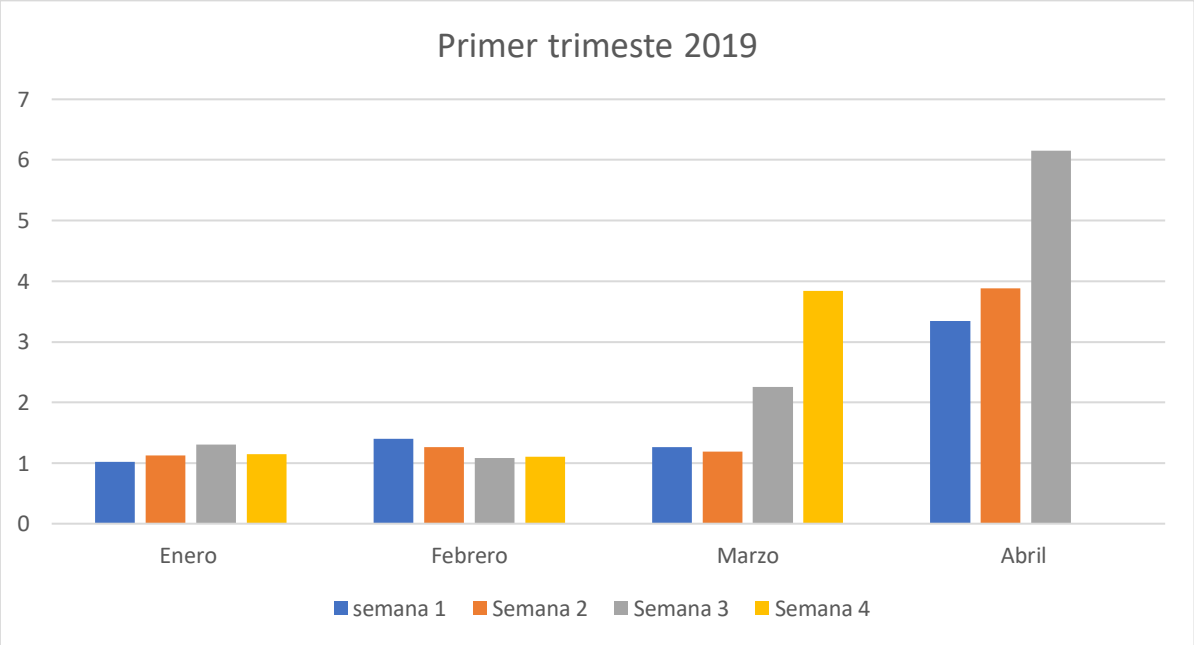


Figura 2. Aumento de personas en la plataforma twich

Hoy en día existe un vasto número de plataformas que ofrecen este tipo de servicio, por ejemplo, las redes sociales que permiten llevar a cabo la transmisión de video en vivo. En la Figura 3 se pueden observar algunas de las plataformas que brindan servicio de video en vivo.

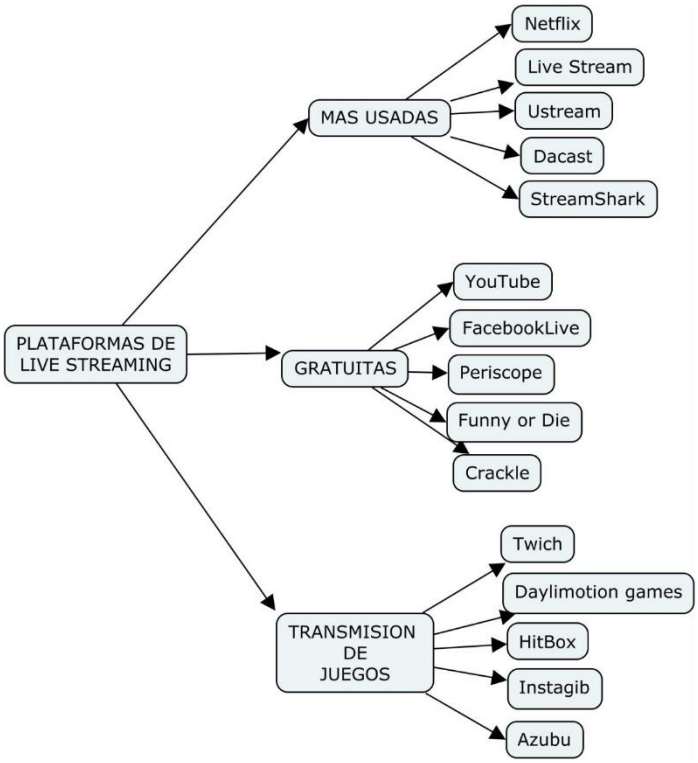


Figura 3. Clasificación de plataformas Live Streaming

Los servicios de *live streaming* pueden ser distribuidos por redes *CDN*. La arquitectura *CDN* (*Content Delivery Network*) cuenta con un servidor principal donde se encuentra alojado el contenido, posteriormente este servidor distribuye el contenido a diversos servidores que se encuentran establecidos en otras partes geográficas, para que los usuarios acceden al contenido desde los diferentes servidores, evitando con esto la congestión en el servidor principal, por esta razón, la *CDN* brinda estabilidad en la transmisión. Sin embargo, este tipo de red implica un costo elevado por el número de servidores involucrados.

Por otro lado, existe una red llamada *P2P* (*Peer to Peer*) en la cual los usuarios involucrados pueden operar como servidores y consumidores al mismo tiempo. En las redes *P2P* los usuarios son conocidos como *peers* ya que no solo descargan el contenido del archivo, sino que también están habilitados para compartirlo; este tipo de redes generan un costo menor debido a que no se emplean servidores fijos, sino que se aprovecha a cada uno de los *peers* y por ende se logra una mayor escalabilidad.

La arquitectura híbrida *CDN-P2P* es una tecnología empleada para el servicio de transmisión de video debido a que ofrece la escalabilidad de las redes *P2P* y la estabilidad de una red *CDN*. Algunos trabajos de investigación han tomado a bien implementar y evaluar esta arquitectura con el fin de probar que son eficientes en la distribución de video a gran escala sobre Internet [2]. En [3] se hace hincapié que la arquitectura *CDN-P2P* ha sido propuesta para sistemas de transmisión, con la finalidad de alcanzar la escalabilidad de una red *P2P*, así como los pequeños retardos y alto rendimiento de una red *CDN*. Por ejemplo, la plataforma *LiveSky* emplea este tipo de arquitectura.

En consideración con lo descrito anteriormente, el desarrollo de este proyecto se enfoca en emplear un sistema híbrido *CDN-P2P* para la transmisión de video en vivo, el cual brinda la estabilidad de una red *CDN* y al mismo tiempo la escalabilidad de una *P2P*.

El resto del documento se organiza de la siguiente forma: el capítulo 1 está orientada a exponer el problema que se va a abordar en el proyecto, así como las limitaciones, justificaciones y objetivos de este. En el capítulo 2 se abordan los trabajos relacionados a este proyecto. En el capítulo 3, se proporciona la teoría necesaria para comprender la terminología empleada. Después en el capítulo 4 se realiza el análisis de problemáticas similares a la abordada en este proyecto. El capítulo 5 proporciona el diseño de la solución a implementar para la cadena de Markov y el diseño del nuevo esquema de asignación de recursos bajo una distribución uniforme. Posteriormente se establecen las conclusiones generales de este trabajo de investigación. Por último, se muestran las referencias utilizadas para desarrollar el presente proyecto.

1.2 Planteamiento del Problema

Actualmente las tendencias de distribución de contenido apuntan al servicio de transmisión de video en vivo como un sector de gran importancia en la vida diaria. Derivado de la situación en la cual se encuentra la sociedad en los últimos años debido a la pandemia Sars-Cov2 (covid 19), el aforo a eventos masivos era limitado para evitar el riesgo de contagio, por lo cual la transmisión de contenido en vivo resulto favorable. Tal es el caso de premiaciones, partidos de fútbol, conciertos, concursos por mencionar algunos. Estos servicios hacen uso de la transmisión en vivo para que el contenido llegue a todos los usuarios que están interesados en este. De igual manera algunas aplicaciones como *Facebook Live, Instagram Stories, Periscope, WhatsApp, Snapchat*, etc. [4], resultan innovadoras y han crecido rápidamente gracias a que transmiten contenido de este estilo.

Sin embargo, cuando se presenta una transmisión de video en vivo desde una plataforma *streaming*, debido a la alta demanda del video por parte de los espectadores, se presenta sobrecarga de peticiones a los servidores donde se aloja temporalmente el contenido, y se ocasiona el efecto cuello de botella en esta parte de la red. Esto afecta la velocidad de flujo y en consecuencia un retardo en la distribución del contenido, ya que no llega al mismo tiempo a todos los usuarios interesados en este. De igual forma, se afecta la experiencia de visualizar el contenido en tiempo real; para lo cual el uso de una red básica cliente-servidor resulta insuficiente [5] ya que no proporciona un alcance adecuado para proveer el servicio a todos los clientes que consumen el video en vivo.

La variedad de aplicaciones multimedia en conjunto con el desarrollo acelerado de las comunicaciones móviles y la accesibilidad a múltiples dispositivos finales permiten emplear el ancho de banda del tráfico de datos presente en las redes subyacentes. [1]

En la literatura existen trabajos que han modelado servicios de video basados en cadenas de Markov. Estos modelos han probado ser muy flexibles y al compararse con otros métodos de modelado han mostrado ser muy precisos. Desafortunadamente, estos desarrollos se han concentrado en servicios de video bajo demanda [5] [6] [7]. Hasta este punto de la investigación se conoce que el modelado con cadenas de Markov aún no ha sido aplicado a servicios de video en vivo.

Cabe mencionar que estos modelos utilizados para *VoD* no pueden ser aplicables directamente a *live streaming* debido a que:

En *VoD* los usuarios tienen poca sincronización entre ellos, puesto que el contenido ya se ha generado con anterioridad y los usuarios no esperan una experiencia en tiempo real. En cambio, en *live streaming*, los usuarios están altamente sincronizados. Como consecuencia, los procesos de arribos en cada caso son significativamente diferentes.

Otra consecuencia de la sincronización en *live streaming* es que los usuarios están interesados únicamente en fragmentos del video de lo que está ocurriendo en tiempo real, mientras que en *VoD* se pueden tener usuarios interesados en intervalos muy diversos del video. Por lo tanto, las poblaciones que pueden compartir recursos son muy diferentes en cada caso.

En *VoD* las tasas de descarga pueden ser significativamente más altas que las tasas de reproducción, porque el contenido ya ha sido almacenado. Lo anterior permite al usuario de *VoD* pre-almacenar gran parte del contenido. Por otro lado, en *live streaming*, este pre-almacenamiento intensivo no es factible, lo que, de nueva cuenta, afecta al modelo e incluso a los esquemas de enfocados en compartir contenido.

En *VoD* el video es estático y por lo tanto las tasas de descarga y reproducción se mantienen de cierto modo constantes, sin embargo, en *live streaming* la tasa de reproducción está altamente ligada con la velocidad a la que se genera el video (ocurre el evento) y por lo tanto el usuario debe reproducir y descargar a esa velocidad para mantenerse en tiempo real.

Debido a que, en la transmisión de video en vivo, el contenido se genera de manera simultánea a su visualización se pueden presentar diversos congelamientos en la reproducción en consecuencia se pierde la *QoE* de visualizar el contenido en tiempo real. Esto se atribuye a múltiples factores, por ejemplo, que no existan los *peers* suficientes para distribuir el contenido a los *peers* que se conecten posteriormente al sistema.

En este caso el servidor principal se ve obligado a proporcionar el contenido a todos los *peers* que arriben al sistema, sin embargo, este hecho puede incrementar el costo de la transmisión de video afectando inherentemente al usuario final. Por otro lado, si el video se pausa de forma continua, los usuarios pierden el interés en visualizarlo y por lo tanto dejan de seguirlo, esto genera pérdidas a las plataformas que distribuyen el contenido.

Ante este escenario se plantea la siguiente pregunta: ¿Cómo desarrollar un modelo, basado en una cadena de Markov, que represente el comportamiento de una población de *peers* que consume el servicio de video en vivo y que además permita el diseño de un nuevo esquema de asignación de recursos?

1.3 Propuesta de Solución

Con este proyecto lo que se busca es modelar el servicio de video en vivo a través de una red híbrida *P2P-CDN* para conocer el comportamiento de los *peers* una vez conectados al sistema y visualizando el contenido de una transmisión en vivo. El video que se trabaja en este modelo es el conjunto de señales de audio y video empaquetados en un solo archivo, debido a que se desea que el usuario tenga la mayor calidad de servicio y experiencia en la visualización del contenido y tomando en cuenta que este es generado en fragmentos llamados *chunks* se propone agrupar estos fragmentos en segmentos más grandes a los cuales denominaremos ventanas.

Esta agrupación se realiza debido a que se plantea que un *peer* pueda descargar una ventana de múltiples *peers* habilitados para compartirle ese contenido, es decir, cuando un *peer* desee descargar una ventana, este podrá descargar fragmentos de esta de diferentes *peers* como se muestra en la Figura 4.

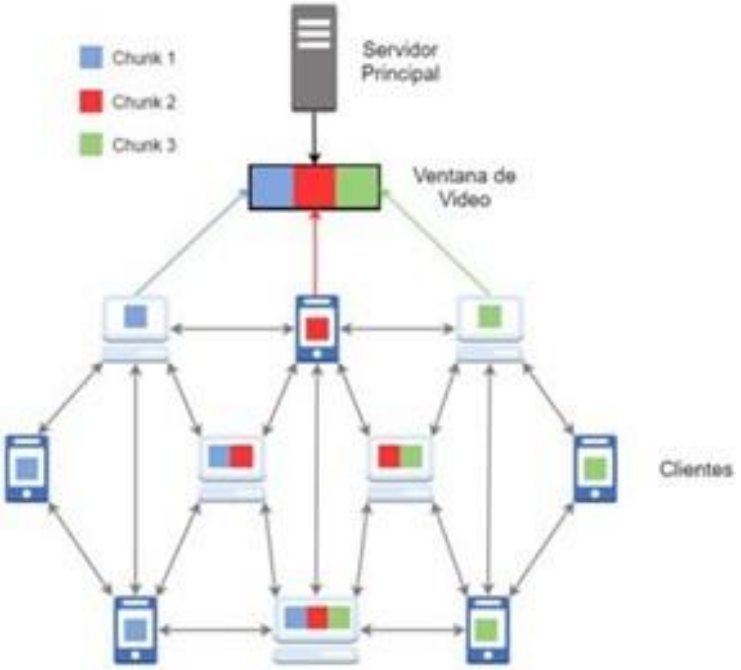


Figura 4. Descarga de una ventana de video

La descarga de las ventanas que componen al video se realizará de acuerdo con el esquema de asignación de recursos que se propone en el presente trabajo ya que con esto se logra reducir el tiempo de descarga y posibles congelamientos en el video.

Una vez que se tiene la segmentación de los fragmentos del video, los usuarios se agruparan de acuerdo con la ventana que se encuentren descargando en ese instante, de esta manera se procede a formar poblaciones de *peers* por ventana de video. Una población de *peers* son los usuarios que están descargando la misma ventana del video en un mismo instante, la población recibe el nombre de acuerdo con la ventana de descarga, es decir, si se encuentran descargando la ventana cero, la población de *peers* en esta ventana se llamara población X_0.

Dentro del sistema desarrollado existen cuatro sucesos principales que tienen ocurrencia dependiente del comportamiento de los *peers* y los cuales afectan directamente al estado de la cadena de Markov. Los sucesos o eventos son los siguientes:

a) La llegada o conexión de un nuevo usuario al sistema, este suceso será cuantificado por una tasa de arribo llamada lambda (λ).
b) Transferencia de un peer a la venta subsecuente, la cual ocurre cuando un *peer* en la población X_i termina de descargar el contenido de la ventana i y pasa a la ventana $i+1$. Este suceso ocurre a una tasa tao(τ)
c) Transferencia de un peer a la venta inferior inmediata, la cual ocurre cuando un *peer* no descarga el contenido de video a la misma tasa que realiza la visualización de este. La captura del evento en tiempo real sigue su curso, por lo tanto, el usuario se atrasa y es transferido a la ventana

inferior. Si está en la ventana i cambiará a la ventana $i-1$, este evento es cuantificado por la tasa (P_ω)

d) Abandono del sistema, este suceso ocurre cuando por alguna circunstancia un usuario sale del evento antes de que esta finalice. Este evento será estimado por la tasa de abandono θ.

Por otro lado, el esquema de asignación de recursos definido en este proyecto tiene como principal objetivo aprovechar al máximo los recursos provenientes de la red *P2P*, para lo cual se establece que los *peers* podrán obtener recursos para descargar el video de *peers* que tengan un mayor progreso en la descarga del contenido.

El diseño del esquema de asignación de recursos será el resultado del análisis de esquemas reportados en la literatura. Del cumulo de esquemas investigados se eligen dos esquemas de asignación de recursos adicionales al esquema de asignación de recursos de distribución uniforme.

En primera instancia el esquema desarrollado se realizará siguiendo el eje de funcionamiento del esquema de asignación de recursos de distribución uniforme, el cual indica que un usuario puede obtener recursos de usuarios ubicados en ventanas superiores a la que este está descargando.

En ese mismo sentido, se obtendrá una expresión de acuerdo con el esquema de distribución uniforme siguiendo los parámetros y limitaciones propios de un servicio de video en vivo. De forma general, este se expresa abiertamente, que el esquema de distribución uniforme no presenta una asignación de recursos óptimas, debido a que los usuarios reciben los recursos en función del tamaño de su población.

Lo descrito anteriormente puede generar que algunas poblaciones de *peers* tengan recursos disponibles de sobra mientras que otras poblaciones se encuentran bajo condiciones de penuria en cuanto a recursos se refiere. A pesar de esto, el análisis basado en la distribución uniforme resulta de gran ayuda para comprender como se puede delimitar el acceso a los recursos de tal forma que el desempeño del sistema sea óptimo.

Además, este análisis facilita la comprensión de la asignación de recursos. Es decir, al estudiar el esquema de asignación de recursos de distribución uniforme y aplicarlo a servicios de video en vivo se comprende la forma en que los recursos son distribuidos dentro de la red y la razón de esta distribución.

De manera preliminar, el diseño del esquema de asignación de recursos para servicios de video en vivo también se realiza tomando como base los esquemas de asignación de recursos: Esquema Q ventanas hacia atrás y DVPG.

A lo largo del desarrollo del presente proyecto y una vez analizados los esquemas de asignación de recursos se establecen las bases que se tomaron para el desarrollo de las expresiones descritas en este documento.

Debido a que la simulación de este tipo de sistemas es realmente compleja y se requiere de diversos equipos tecnológicos para validar los resultados el proyecto se limita a evaluar matemáticamente la cadena de Markov que representa al sistema y a evaluar el esquema de asignación de recursos propuesto para servicios de video en vivo. Dichas evaluaciones se realizarán tomando parámetros de operación utilizados en sistemas de distribución de video reales.

1.4 Alcances

Este proyecto se limita a definir una cadena de Markov que represente el comportamiento de los usuarios a lo largo de una transmisión de video en vivo a través de una red hibrida *P2P*-CDN. En dicha cadena se observa de igual forma la relación que tienen los usuarios con el archivo de video y la composición de este. Finalmente, en la cadena se muestran los eventos que generan un cambio en su estado al momento de ocurrir, así como la tasa a la cual ocurre alguno de estos eventos.

El desarrollo de la cadena se realiza de forma cuasi paralela al diseño de un nuevo esquema de asignación de recursos. En primera instancia se considera al esquema de asignación de recursos tradicional uniforme para plasmar el comportamiento de los usuarios y la forma en la cual estos consumen recursos para poder descargar y visualizar el video en vivo.

Posteriormente se plantea trabajar la cadena definida bajos el esquema uniforme así mismo en el nuevo esquema que se propone en el presente trabajo. Dicho esquema se diseña de acuerdo con esquemas de asignación de recursos que se han investigado hasta el momento, pero con la diferencia de abstraer el método que emplean para asignar los recursos y adaptarlo al concepto de *live streaming*.

El estado de la cadena de Markov definida para servicios de video en vivo considera que un instante de tiempo ocurre un único evento por parte de los usuarios (conexión, desconexión o transferencia).

Las implementaciones de la cadena de Markov y la evaluación del nuevo esquema de asignación de recursos que se propondrá en este proyecto se realizarán empleando parámetros (tasa de subida, tasa de descarga, tasa de desconexión, etc.) que se han reportado en trabajos de investigación relacionados a los servicios de video y los esquemas de asignación de recursos.

La solución matemática de la cadena de Markov y del esquema de asignación de recursos se implementan en el ambiente de desarrollo Matlab, con la finalidad de obtener la cantidad de usuarios promedio situada en cada una de las ventanas que componen al video.

Estos resultados son traducidos a tasas de subida y descarga requeridos por el sistema para la distribución de contenido en vivo y de esta forma conocer el desempeño general y las condiciones que presenta el sistema.

Debido a que un archivo de video se produce por pequeños *frames,* a los cuales se les denomina *chunk* y estos se agrupan en ventanas, un usuario conectado al sistema, conocido a partir de su conexión como *peer*, podrá obtener *chunks* de múltiples *peers* que tengan la capacidad de compartir el contenido.

El sistema que se desarrolla considera que, al momento de la conexión, un *peer* visualizará el contenido en baja calidad hasta que se obtenga parte significativa del archivo de video, con el propósito de que se cree *buffer* y evitar congelamientos en la reproducción o estancamiento en la descarga del video.

Este proyecto no considera la implementación de un simulador para este modelo, sin embargo, se presentarán las estadísticas de consumo y desempeño del sistema como ya se ha mencionado en función de parámetros en sistemas reales.

1.5 Justificación

A raíz de la pandemia por SARS-CoV-2 el consumo de servicios de video en vivo presento un incremento respecto al consumo de *live streaming* en años anteriores. Actualmente múltiples plataformas de distribución de video y redes sociales disponen de este servicio para compartir eventos del día a día en tiempo real. Este servicio permite tener acceso a contenido generado de forma simultánea a su distribución dentro de la red, desde cualquier ubicación geográfica del planeta. *Live streaming* es un servicio que se distribuye en redes del tipo tradicional (cliente-servidor) y redes híbridas (*P2P-CDN*).

Existen diversos modelos para servicios de video bajo demanda (*VoD*), sin embargo, no se tiene conocimiento acerca de un modelo que represente los servicios de video en vivo. Por esta razón es que se ha decido realizar el modelado de este tipo de servicio mediante cadenas de Markov. El análisis de los eventos, sucesos y comportamiento por parte de los usuarios en el consumo de *live streaming* es análogo a lo que sucede en el ambiente del video bajo demanda, pero con la diferencia de que *live streaming* es un archivo dinámico y no estático como el caso de *VoD*.

Por otra parte, también existen múltiples esquemas de asignación de recursos aplicados a *VoD* y no se conoce algún esquema propio para video en vivo. En este proyecto se han resaltado los esquemas con un óptimo desempeño para servicios de video bajo demanda con el fin de conocer los parámetros que emplean para la distribución de recursos entre los usuarios y analizar el desempeño que presentan. Este análisis permite desarrollar un esquema de asignación de recursos funcional con un óptimo desempeño para servicios de video en vivo.

Por lo descrito anteriormente es que se decide desarrollar en este proyecto el modelado de servicios de video en vivo mediante una cadena de Markov y desarrollar un esquema de asignación de recursos propio para los servicios de video en vivo sobre una red híbrida *P2P-CDN*. El modelado mediante cadenas de Markov permitirá conocer el comportamiento de los usuarios al momento de consumir un servicio de video y su repercusión en la dinámica del sistema que lo transmite.

A su vez este modelo permite conocer las tasas de operación del sistema y su relación con el desempeño general del mismo. Así mismo, se podrán obtener estadísticas de consumo y desempeño del sistema con la finalidad de buscar que estas sean óptimas.

Finalmente se busca que el esquema de asignación de recursos realice la distribución de estos de forma equitativa entre los usuarios que consumen el video en vivo, para que estos puedan visualizar el contenido con buenas *QoE* y *QoS*. Por otro lado, este esquema también busca obtener mayor proporción de recursos de la red *P2P* y disminuir el consumo de recursos provenientes de la red *CDN*.

1.6 Metodología

La metodología que se optó implementar para el desarrollo del proyecto es *SCRUM*, debido a que esta se caracteriza por ser una metodología basada en una estructura de desarrollo incremental, lo cual quiere decir, que cualquier ciclo de desarrollo del producto y/o servicio se descompone en pequeños proyectos divididos en distintas etapas: análisis, desarrollo, implementación y evaluación. En la etapa de desarrollo se realizan interacciones del proceso o *sprint*, es decir, entregas regulares y parciales del producto final.

De acuerdo con la complejidad de este proyecto y el número de desarrollados, esta metodología se adapta de forma adecuada ya que permite abordar

proyectos complejos que exigen una flexibilidad y una rapidez esencial al momento de ejecutar los resultados.

SCRUM está orientada a gestionar y normalizar los errores que se puedan producir en procesos de desarrollo de larga duración, a través de, reuniones frecuentes para asegurar el cumplimiento de los objetivos establecidos. Para este proyecto esta característica ha sido fundamental al momento de programar reuniones para debatir conceptos, establecer técnicas de trabajo, definir objetivos e intercambiar experiencias personales y de esta forma fomentar la proactividad.

Finalmente, *SCRUM* aporta al proyecto: innovación, flexibilidad, competitividad y productividad.

Siguiendo la metodología antes menciona se han trazado las siguientes fases para el desarrollo de este proyecto:

Análisis

En esta fase del proyecto se realizaron las siguientes actividades:

- Investigación sobre los conceptos clave de archivos de video
- Investigación sobre las cadenas de Markov
- Investigación sobre conceptos de arquitecturas de red
- Investigación acerca del modelado de servicios de video mediante cadenas de Markov
- Investigación de esquemas de asignación de recursos
- Análisis de modelos para servicios de video bajo demanda (*VoD*)
- Análisis de esquemas de asignación de recursos
- Análisis de una cadena de Markov unidimensional

Con base en las actividades antes mencionadas se redactaron las siguientes secciones del presente documento: Introducción, planteamiento del problema, propuesta de solución, alcances, objetivos y marco teórico.

Diseño

En esta etapa del proyecto se realizaron las siguientes actividades:

- Diseño de la cadena de Markov
- Diseño de la solución matemática de la cadena
- Bosquejo del nuevo esquema de asignación de recursos

Para el diseño de la cadena de Markov que representa los servicios de video en vivo distribuidos sobre una red híbrida *P2P-CDN* se definieron en primer lugar los eventos (comportamiento de los *peers*) que afectan directamente al estado de la cadena y producen un cambio en este.

Posteriormente se definió la estructura que debe tener el archivo de video y la clasificación de los *peers* una vez que están conectados al sistema. Además, se estableció la nomenclatura que representa las tasas de ocurrencia de un evento por parte de los usuarios.

Finalmente se identificaron los cambios que debe tener el estado de la cadena de Markov una vez ocurrido un evento.

En el diagramado de la solución matemática se describió en forma de diagrama de flujo el procedimiento que se debe seguir para implementar la cadena de Markov, es decir, se estableció el algoritmo que se deberá implementar para simular la ocurrencia de eventos por parte de los usuarios dentro de un sistema de video en vivo para poder obtener estadísticas de consumo y desempeño de este basado en las transiciones del estado de la cadena.

Para el bosquejo del nuevo esquema de asignación de recursos se analizaron distintos esquemas de asignación de recursos enfocados al servicio de video bajo demanda y se identificaron las principales diferencias entre este servicio y el de *live streaming* para de esta forma poder establecer un esquema a fin para el servicio de video en vivo.

Implementación

En esta etapa del proyecto se desarrollaron las siguientes actividades:

- Implementación del diagrama de flujo de la solución matemática de la cadena de Markov: esta actividad se refiere a la implementación de los eventos por parte de los usuarios (conexión, desconexión o transferencia) en un IDE que permitió obtener estadísticas de consumo y desempeño significativas para la solución de la cadena y el diseño de un nuevo esquema de asignación de recursos.
- Evaluación del nuevo esquema de asignación de recursos
- Evaluación del esquema propuesto en el proyecto frente a esquemas desarrollados en otros trabajos de investigación.
- Corrección de errores de implementación como lógica de programación y sintaxis.

Evaluación

Finalmente, en esta etapa, se realizarán pruebas de las implementaciones de la cadena de Markov y el nuevo esquema de asignación de recursos para obtener estadísticas del funcionamiento del sistema de servicio en vivo. De acuerdo con los resultados obtenidos en esta etapa se realizaron modificaciones, en caso de ser necesarias, para obtener resultados que permitan al usuario final visualizar el contenido en vivo con *QoE* y *QoS* optimas.

1.7 Objetivos

1.7.1 Objetivo General

Desarrollar un modelo matemático basado en una cadena de Markov, con la finalidad de diseñar un esquema eficiente de asignación de recursos para un sistema de video en vivo, soportado por redes híbridas CDN-P2P.

1.7.2 Objetivos Específicos

Plantear una cadena de Markov basada en las conexiones y desconexiones de los *peers* de cada una de las ventanas del video, para modelar el comportamiento de los múltiples usuarios en el sistema.

Evaluar la cadena de Markov dentro del modelo híbrido CDN-P2P mediante la implementación de métodos numéricos, con la finalidad de estimar la demanda de descarga y la cantidad de recursos de subida disponibles en el sistema.

Diseñar y evaluar un nuevo esquema de asignación de recursos, basado en la solución de la cadena de Markov, con el propósito de mejorar el desempeño del esquema de asignación uniforme tradicional.

Capítulo 2

Estado del Arte

En este capítulo se presentan diversas investigaciones relacionadas al análisis de servicios de video, de manera particular servicios de video bajo demanda *(VoD)*, y al análisis de esquemas de asignación de recursos dentro de un sistema para este tipo de servicios.

2.1 Estado del arte del modelo de servicios de video con cadenas de Markov

En esta sección se introducen investigaciones que muestran gran relación con el análisis de servicios de video bajo *demanda (VoD) mediante* cadenas de Markov y que son tomados como base para el desarrollo del análisis de servicios de video en vivo mediante cadenas de Markov en el presente proyecto. Esto debido a que hasta este momento no se han encontrado en la literatura modelos basados en cadenas de Markov para servicios de video en vivo.

En [8] los autores rescatan las principales características de las redes *CDN* y *P2P* de manera individual y de forma híbrida. En este trabajo se realiza un modelo híbrido *CDN-P2P* sobre la web con la finalidad de disminuir las peticiones realizadas a los servidores *CDN*; se menciona que el *streaming* de video en vivo es una tecnología que permite a un usuario denominado emisor generar un video que se transmite a otros usuarios en tiempo real.

En este trabajo se mencionan algunas ventajas de las redes *CDN* y *P2P*. De la *CDN* explican que tiene una excelente calidad en el servicio y costos de implementación y mantenimiento relativamente altos. Por otro lado, *P2P* tiene una mejor escalabilidad y menores costos de implementación. El modelo propuesto en este trabajo tiene 2 capas fundamentales, la capa *CDN* y la capa *P2P*. En ese modelo el usuario final se sitúa directamente en la capa *P2P* y en caso de no encontrar los recursos necesarios en su nivel los solicitará al nivel de arriba (*CDN*).

En [9] se compara y evalúa el rendimiento de la red *CDN* frente a la red *P2P*; el punto clave de la *CDN* es replicar el contenido del servidor original en un cache local y de ahí distribuir el contenido a los clientes. Mientras que en la arquitectura *P2P* la distribución se aplica del lado de la red del cliente, ya que estos se convierten en asociados activos al proporcionar el contenido que están recibiendo a otros clientes.

Los trabajos relacionados a *CDN, P2P y CDN hybrid P2P* basados en video en vivo se enfocan en problemas específicos como: reducir el retraso de inicio, distorsión de la transmisión, perdida de paquetes y costo de ancho de banda.

En [10] los autores desarrollan un modelo de transmisión de video sobre una red híbrida *CDN-P2P* para lograr estabilidad y escalabilidad en la difusión de contenido. Aunado a esto mencionan que el modelo cuenta también con otros componentes (*SDN, NSP, VCP*) que permiten tener mejores parámetros en la visualización, la experiencia y el servicio. El modelo al ser basado en *SDN* permite que todos los usuarios puedan aprovechar los recursos disponibles en

la *CDN-P2P* de forma inteligente para tener mejor calidad de experiencia y de servicio. Por otro lado, con *NSP* y *VCP* se logra tener mayores recursos en la periferia de un usuario y con ello se logra un mejor rendimiento en la red. De igual forma en este trabajo los autores mencionan los algoritmos empleados para clasificar a los *peers* en diversos grupos y con esto verificar que los recursos sean asignados equitativamente.

En [11] se proporciona un sistema de transmisión hibrido (Video en vivo y *VoD*), basado en codificación de video escalable *SVC* (*Scalable Video Coding*). El sistema consiste en un transcodificador en línea *SVC* diseñado para soportar otros formatos de video y transcodificarlo a *SVC*. En este trabajo se propone un servidor de *streaming* basado en *SVC* para hacer paquetes tipo *RTP* y distribuirlos dentro de la red. El servidor *P2P* recibe los paquetes desde el servidor *streaming* y se encarga de distribuirlos a los *peers* (empleado en el sistema en vivo). Además, existe un *tracker* con el fin de mantener la información y el estado de todos los *peers*, para clasificarlos en *seed peer* y *common peer* dependiendo de la carga y descarga que estén realizando. Finalmente se menciona que el reproductor *SVC* únicamente está disponible para PC's desarrolladas como un complemento en ActiveX y teléfonos inteligentes. Con este modelo se logra reducir la probabilidad de perdida y retraso de paquetes.

En [7] se establece un modelo para la distribución de video bajo demanda en redes heterogéneas 5G. En este trabajo se aborda un modelo fluido (sistema de ecuaciones) que representa el comportamiento de los *peers* conectados al servicio. Para analizar el comportamiento de los *peers* se tomaron en cuenta parámetros como duración del video, ancho de banda de los servidores, tiempo de permanencia, entre otros. Estos datos permitieron evaluar el esquema para observar si la asignación de recursos se realizaba de forma uniforme y equitativa para todos los usuarios. Los resultados fueron comparados con resultados obtenidos en otras investigaciones que utilizan cadenas de Markov para analizar el compartimiento de los *peers* en esos sistemas. De igual forma en este trabajo se establece que sí el modelo es exitoso se puede generar un modelo que reduzca el número de recursos utilizados para su funcionamiento. Los resultados que se obtuvieron se consideran veraces por la similitud que tienen con los resultados basados en cadenas de Markov de otras investigaciones basadas en el funcionamiento real que muestra la plataforma de distribución de contenido YouTube.

En [6] los autores presentan un *framework* analítico para modelar el servicio de video bajo demanda (*VoD*) a través de una red híbrida *CDN-P2P*, su trabajo está basado principalmente en un modelo fluido y en cadenas de Markov. El modelo fluido permite a los autores comprender el comportamiento del sistema en términos de retrasos por parte de los usuarios, congelamiento en la transmisión del contenido, desfase en la reproducción de una ventana y la descarga de la ventana posterior, la clasificación de los *peers*, etc. Por su parte la cadena de

Markov se emplea para conocer la tasa de subida y bajada que presenta el sistema, es decir, la cantidad de *peers* que se encuentran en cada grupo (j, k), lo que quiere decir que dichos *peers*, están reproduciendo la ventana j y descargando la ventana k. Al final del trabajo los autores muestran los resultados numéricos obtenidos basándose en diferentes parámetros *QoS*.

En [12] el autor propone un nuevo esquema de asignación de recursos llamado Distribución por Ventanas Priorizadas (DVP). El esquema propuesto en este trabajo demostró ser más eficiente comparado con otros esquemas de asignación de recursos, esto al reducir el ancho de banda solicitado a los servidores. Este trabajo se enfoca en analizar un archivo de video segmentado en ventanas de igual tamaño (red heterogénea) mediante un modelo fluido y complementar con un análisis mediante cadenas de Markov. Para realizar dichos análisis el autor se basa en diversos parámetros de *QoS* y *QoE*. Cabe mencionar que el video analizado es un video bajo demanda (*VoD*) a través de una red *P2P*. Finalmente el autor reporta los resultados obtenidos a partir de diferentes parámetros *QoS* y *QoE* de entrada.

En [5] se menciona que las redes *P2P* son una tecnología clave para la distribución de contenido de video dentro de las próximas generaciones de comunicaciones inalámbricas, incluyendo la quinta generación de sistemas móviles denominada 5G. En este trabajo se fragmenta el contenido de video en ventanas que a su vez están compuestas por partes más pequeñas llamadas *chunks*, además se dice que el tamaño de la primera ventana puede ser diferente a la de las demás que componen el archivo de video, con la finalidad de obtener un equilibrio entre el retraso inicial y la duración de las pausas.

2.2 Estado del arte de esquemas de asignación de recursos para servicios de video

En esta sección se presentan trabajos de investigación en los que se ha abordado la asignación de recursos para servicios de video bajo demanda *(VoD)* con la finalidad de conocer el desempeño y funcionamiento de dichos esquemas.

En [7] el autor menciona que los recursos disponibles en una red deben ser asignados correctamente para obtener un buen desempeño del sistema para la descarga y visualización de un video. En este trabajo se implementa una red heterogénea, lo cual indica que todos los usuarios conectados a la red tendrán la misma oportunidad de acceder a los recursos disponibles. Sin embargo, esto implica que los recursos se distribuyen de acuerdo con el esquema de asignación de recursos uniforme.

Por esa razón en este trabajo el autor propone un esquema de asignación de recursos diferente, al cual llamó esquema de Q ventanas hacia atrás, este esquema establece que todos los *peers* que deseen descargar y reproducir un video tendrán la misma oportunidad de consumir recursos de *peers* en Q ventanas hacia adelante y del servidor principal. Con esto se garantiza una correcta distribución de recursos y se evita que algunos usuarios se encuentren en sobre abundancia mientras que otros están en penuria. El parámetro llamado Q, es definido como la cantidad de ventanas superiores desde donde un usuario podrá consumir recursos.

En [13] el autor menciona las ventajas que ofrece una red híbrida *CDN-P2P* para la transmisión, descarga y reproducción de un video bajo demanda *(VoD)*. En este tipo de redes los usuarios, llamados *peer* toman el papel de servidor y consumidor al mismo tiempo, con lo cual se incrementa la cantidad de recursos disponibles para descargar y reproducir un video. En este trabajo se realizan diversas comparaciones de esquemas de asignación de recursos con el fin de obtener puntos de mejora y así proponer un nuevo esquema de asignación de recursos. Como resultado de esta investigación se tiene un esquema de asignación de recursos que tiene como parámetro a Q, que es la cantidad de ventanas hacia arriba que se tomarán en cuenta para obtener recursos, es decir, si un *peer* desea descargar la ventana i, los usuarios que le podrán proporcionar recursos son los usuarios que estén descargando la ventana j siempre y cuando $\mathrm{i} < \mathrm{j} < \mathrm{i} + \mathrm{Q}$. Por otro lado, este esquema es iterativo, es decir, si un video está constituido por N ventanas, el parámetro Q deberá cambiar en función de $i + Q < N - 1$.

En [6] el autor hace una evaluación de diferentes esquemas de asignación de recursos presentes en la distribución de contenido sobre redes *CDN, P2P* y redes híbridas *CDN-P2P* con el fin de obtener el desempeño que estos esquemas muestran frente a la distribución de video bajo demanda y así proponer un nuevo esquema de asignación de recursos con base en el modelo fluido que describe el autor, en el cual se explica el comportamiento de los *peers* y las reglas que se deben seguir para una correcta descarga y reproducción del video.

El autor menciona que en diversos esquemas sólo basta con que un *peer* desee descargar el contenido de una ventana i y el grupo de *peers* en ventanas superiores, es decir, en ventanas $\mathrm{j} > \mathrm{i}$ le proporcionarán recursos, sin embargo, también menciona que esta práctica inevitablemente genera una disparidad y provoca que algunos *peers* tengan más recursos disponibles para consumir que otros. En el esquema que propone, los recursos provenientes de los servidores *CDN* y de los demás *peers* conectados al sistema se asignan de manera equitativa. En este esquema se establece un parámetro de prioridad denotado por ε, el cual controla la cantidad de recursos asignada a un grupo de *peers* perteneciente a una ventana i. Finalmente el autor establece que este esquema es mejor en comparación con el esquema uniforme y el esquema PWA.

En [12] el autor propone un nuevo esquema de asignación de recursos llamado Distribución por Ventanas Priorizadas (DVP). La idea principal del esquema propuesto en este trabajo es asignar recursos con mayor prioridad para la descarga de las partes del video que se encuentran en penuria, específicamente a las ventanas altas.

Dentro de la asignación de los recursos por priorización de ventana existen dos tipos de recursos: los provenientes de los servidores y los provenientes de los *seed*s. El uso de DPV para asignar los recursos de los *downloaders* provee un esquema más flexible y eficiente ya que propicia que los *peers* en una ventana en específico aprovechen los recursos de *downloaders* en ventanas inmediatas superiores.

En este esquema los recursos de los *downloaders* en la ventana i son inútiles para *peers* con progresos de descarga mayores, es decir, situados en ventanas $j,\ j > i$. Finalmente, el autor realiza una generalización del esquema DPV y lo llama DVPG, esto con el fin de analizar de forma conjunta los recursos provenientes tanto de *seed*s como de *downloaders (peers)*.

En [14] el autor analiza un archivo de video bajo demanda que es dividido en ventanas y consumido por diversos usuarios conectados al sistema. Estos usuarios reciben el nombre de *peers* y requieren consumir recursos para descargar y reproducir el video. En este artículo el autor propone un esquema de asignación de recursos llamado *INUA* (*Immediate-neighbors Uniform Allocation*).

El esquema *INUA* establece que un *peer* podrá obtener recursos de *peers* con un progreso de descarga mayor, es decir, un *peer* que se encuentra descargando la ventana j, podrá obtener recursos de usuarios en ventanas $i,\ i > j$. Al conjunto de ventanas con mayor progreso de descarga se le llama Uj y es de longitud Mj. Por lo tanto, las ventanas i que pueden proporcionar recursos están acotadas por los limites $j + 1 < i < j + Mj$.

Capítulo 3

Marco Teórico

En esta sección se describe al video en vivo, así como los diversos formatos empleados para la codificación del contenido. De igual forma, se muestran los formatos utilizados para el empaquetamiento de este. Enseguida se muestran los protocolos para la distribución de video en vivo por Internet. Y finalmente se enuncia la manera en la cual operan las redes híbridas *CDN-P2P* en los servicios de video en vivo.

3.1 Video en Vivo

3.1.1 Formatos de codificación

En la transmisión de contenido en vivo se usa como principal herramienta a un *códec*, que es el acrónimo de codificador-decodificador. Esto se debe a que la señal de video necesita ser codificada en un formato de *streaming* que permita tener las menores pérdidas de calidad y sonido posibles. Los *códecs* realizan la codificación *streaming* para comprimir el contenido a emitir y descifrarlo una vez recibido en los diferentes dispositivos móviles para la reproducción.

A continuación, se describen algunos de los formatos de codificación que permiten la transmisión de audio y video en un solo paquete a través de una red. Información obtenida a partir de [15] [16].

Mjpeg: Un video no es más que una sucesión de imágenes en movimiento. Si se comprimen todas las imágenes en formato JPEG se obtendría el formato MJPEG, o Motion JPG. Con este formato se logra una buena compresión con respecto al original.

H264: es un formato de codificación de video para grabar y distribuir señales de video FullHD y audio. Fue desarrollado y mantenido por el ITU-T Video Coding Experts Group (VCEG) con el ISO/IEC JTC1 Moving Picture Experts Group (MPEG). Empleado normalmente para grabación, compresión y distribución de contenidos de vídeo, el formato H.264 es un método de transmisión de video compatible con redes de datos, que suministra imágenes de alta calidad sin consumir demasiado ancho de banda.

Mpeg4: Es una evolución de MPEG. Es más avanzado y ofrece numerosas opciones de configuración y muy alta calidad. Sin embargo, su edición exige gran potencia en hardware.

VP8 (Formato de Compresión de Video o Especificación de Compresión de Video): Es una especificación para codificar y decodificar video de alta definición tanto en archivo como en flujo de bits para visualizar.

VP9: Es de código abierto y libre de regalías. Fue desarrollado por *Google* como sucesor del VP8, la alternativa moderadamente exitosa al H.264. Durante su desarrollo VP9 fue apodado “NGOV” (*Next Gen Open Video*) y *Google* ya ha integrado el soporte en el navegador *Chrome* y en *YouTube*.

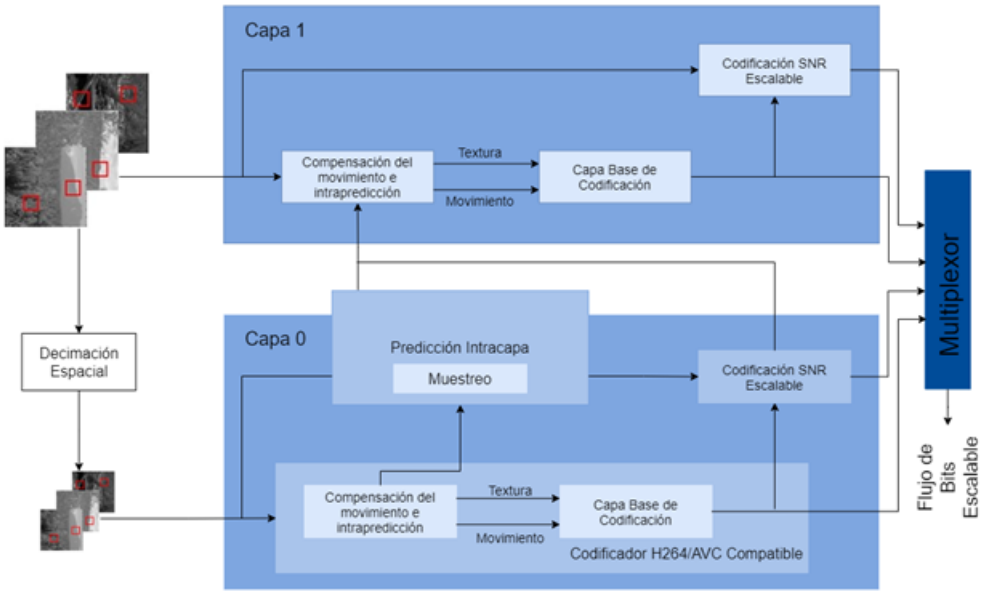


Figura 5. Diagrama a bloques del códec H264 [17]

3.1.2 Formatos de empaquetamiento

A continuación, se mencionan los formatos de empaquetado más usados dentro de la distribución de video y audio en vivo [18]

.avi: Es el empaquetado multimedia con mayor utilidad como archivo de vídeo. Significa audio video interleave, es decir video y audio entremezclados.

.mov: Empaquetado por defecto de *QuickTime* y productos *Apple*. Otras extensiones recientes compatibles con .mov son: .mp4, .m4a, .3gp, .mj2.

.mpeg: Empaquetado ampliamente utilizado, por ejemplo, en la emisión TDT. Otras extensiones compatibles son .MOD y .mpg.

.vob: Formato mpeg2 con opción de encriptado para su uso en el DVD.

.flv: Formato propio de las animaciones *flash.*

.ogg: Empaquetado tradicionalmente usado para el audio vorbis.

.mkv, .matroska: Empaquetado especial para alta densidad de datos en HD.

.webm, .WebM: Formato de empaquetado libre de última generación

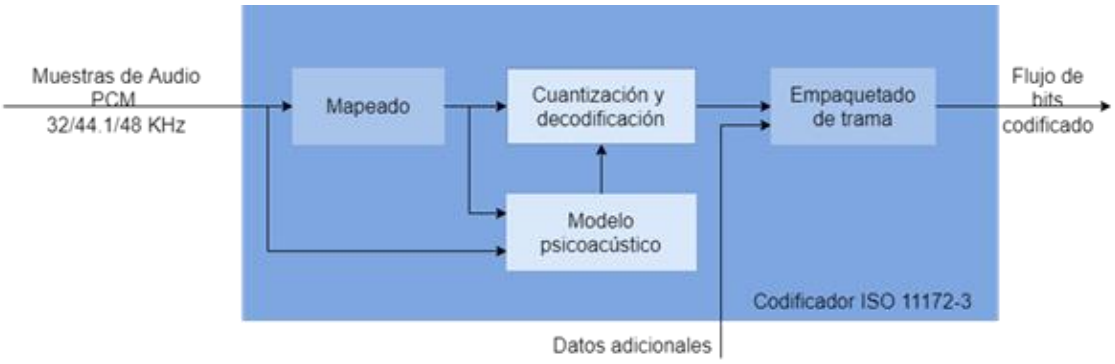


Figura 6. Diagrama a bloques del formato de codificación mpeg [19]

3.1.3 Servicios de video en vivo

Un *stream* de contenido multimedia, generalmente llamado video, está compuesto por varios *streamins* simples que poseen un tipo de contenido especifico (audio, video o subtítulos). Cada *stream* simple es codificado por un *códec* en un formato especifico como h264, mp4 o mjpeg para el caso de la señal de video. En el caso de la señal de audio existen formatos como Mp3, aac, ogg, pm_s16le.

Los contenidos de un *stream* de multimedia son empaquetados en un nuevo formato llamado *container*, mov, avi, mpeg,flv, vob son algunos de estos. El objeto de emplear un *container* es lograr sincronía entre los contenidos. Una vez realizado el empaquetado el *stream* de multimedia se puede almacenar como archivo en un servidor o se puede transmitir en directo.

Los archivos de video comúnmente son producidos y distribuidos por una amplia gama de equipos de usuario (UE), tales como teléfonos celulares, dispositivos inteligentes, o dispositivos conectados al Internet de las cosas (*IoT*). Aunado a esto, durante el proceso de producción de videos, se pueden encontrar diferentes tipos de contenido de video, como películas, anuncios, aplicaciones de realidad aumentada, videos de seguridad, etc. En la Tabla 1 se enlistan las características principales de los archivos de video y en la Tabla 2 las características principales de un video en vivo.

Atributo	Descripción
Espacio de Color	Modelo matemático que asigna una representación del color a su equivalente perceptivo.
Resolución	El número de píxeles en una ventana de video, expresada como $N * M$ píxeles, donde N y M representan el número de columnas y filas respectivamente, en la matriz de píxeles.

Proporcionalidad	Relación entre el ancho y alto de la ventana de video.
Tasa de Ventana	El número de Ventanas por segundo(fps) de un video.
Tasa de bit	El número de bits necesario para representar un segundo de un video codificado.
Calidad de Video	Fidelidad de un video codificado respecto a su versión original, la cual es medida subjetivamente o mediante una métrica objetiva (PSNR, VQM y VMAF)
Esquema de Codificación	Formato de codificación de video y parámetros de codificación.

Tabla 1. Atributos de los videos [20]

Característica	Descripción
Tiempo real	El evento es capturado a través de un dispositivo y transmitido por la red al mismo tiempo.
Interacción en tiempo real	En caso de que los usuarios puedan realizar preguntas y/u observaciones, estas ocurren a lo largo de la transmisión en vivo.
Sentido de pertenencia	La audiencia espectadora experimenta sensación de exclusividad o pertenencia al grupo que tiene acceso al contenido.
Base para *VoD*	El video capturado en tiempo real se convierte en *VoD* si se almacena y los usuarios acceden a este tiempo después de su grabación.
Edición	El video capturado no tiene ediciones gráficas o de sonido, es decir, se transmite tal cual se graba.
Manejo	Los usuarios no pueden manipular el video en cuanto pausarlo, adelantarlo o retrasarlo se refiere.
Fases	Consta de 3 fases principales: grabación, procesamiento y distribución.

Tabla 2. Características del video en vivo [21] [22]

Los servicios de video distribuidos a través de la Internet son agrupados en dos principales categorías, los servicios bajo demanda y los servicios de *live streaming*.

La distribución de contenido multimedia hace referencia al proceso por el cual se distribuyen los servicios, como películas, clips de video y/o transmisiones en vivo sobre una red, ya sea en tiempo real o no. Existen 2 métodos principales de dispersión de contenido multimedia sobre una red, *downloading* y *streaming*. Por

objetivos del proyecto el *streaming* y una parte de *downloading marcarán uno de los ejes principales de investigación*. [4]

Downloading (*progressive downloading*): es el procedimiento donde el usuario puede visualizar el contenido multimedia mientras lo está recibiendo de otro dispositivo.

Streaming: a diferencia de *downloading*, en este método de distribución se cuenta con un servidor *streaming* especializado, el cual distribuye el contenido multimedia a los diversos usuarios conectados. La distribución se realiza considerando los recursos específicos por asignar a cada uno de los usuarios en función de su solicitud. En este método no es necesario que el archivo de video sea descargado en el dispositivo final del usuario.

El presente proyecto está enfocado en el servicio de *live streaming.* Plataformas como *Sling TV, DirecTV Now, YouTube TV, y Live TV de Hulu* [23] son las encargadas de transmitir contenido en vivo.

3.1.4 Protocolos de arquitectura y distribución de video en vivo por internet

En la transmisión de video en vivo diversas tecnologías trabajan colectivamente para lograr la distribución del contenido sobre una red. Esto se debe a que el *streaming* implica diversas etapas, en primer lugar, se capta el contenido desde el dispositivo donde se produce, posteriormente este contenido entra en la etapa de codificación para así poder ser almacenado y transmitido a los diversos espectadores de manera simultánea.

Un protocolo de transmisión es la tecnología empleada para transportar archivos de video a través de Internet. [24]

Anteriormente el video disponible en la red era entregado por el protocolo RTMP, que es el protocolo de mensajería en tiempo real. Este protocolo es considerado como un estándar basado en *Flash* para *live streaming*. Actualmente aún es utilizado para el envío de video desde el codificador RTMP a una plataforma de video en línea.

Sin embargo, el protocolo RTMP basado en *Flash* ya no es apropiado para entregar video a los usuarios. Esto gracias a que el complemento *Flash* ha perdido importancia debido a que los dispositivos que admiten este protocolo son cada vez menos. En los últimos diez años el protocolo RTMP ha sido reemplazado paulatinamente por el protocolo HLS [24]. De forma paralela el protocolo MPEG-DASH ha tomado un lugar importante en *live streaming*.

MPEG-DASH es el protocolo de transmisión utilizado con mayor frecuencia en los últimos días. Este protocolo fue creado como una opción frente a la fragmentación que sufrió el mercado de transmisión de video causada por la competencia de HLS de *Apple* con diversos protocolos de transmisión de video. Esto obligo a las organizaciones de estándares a desarrollar MPEG-DASH como un protocolo de transmisión estándar de código abierto, al igual que el protocolo de transmisión HLS.

MPEG-DASH es un método de tasa de bits adaptativo para la transmisión de video. Por otro lado, ha mostrado ser compatible con la publicidad y la tecnología, razón por la cual debe estar a la vanguardia. MPEG-DASH es compatible con DRM, entrega HTTP, transmisión de baja latencia y otras características de *live streaming*.

Dispositivos *Android, iOS, Windows, Mac, Linux, Chrome OS* por mencionar algunos, trabajan con el protocolo HLS y MPEG-DASH. Sin embargo, este último no es compatible con el navegador *Safari* móvil, por lo tanto, no es admitido por la mayoría de los usuarios *iPhone, iPad y AppleTV* ya que utilizan el navegador *Safari* como buscador predeterminado.

HLS y MPEG-DASH son protocolos de velocidad de bits adaptables. Lo cual se traduce, en que los usuarios reciben automáticamente el video con mejor calidad a razón de la calidad de conexión a Internet que poseen. En un escenario ideal este hecho proporciona una experiencia de visualización estable y de alta calidad a los espectadores mientras que se reduce el almacenamiento en búfer y el retraso.

A medida que el tráfico de video en las redes celulares crece exponencialmente, los operadores de redes móviles (ORM) aplican tecnologías de quinta generación (5G) de redes de comunicaciones. Esto con la finalidad de satisfacer los requisitos de calidad de servicio y de experiencia en las aplicaciones multimedia. El objetivo principal de la implementación de estas tecnologías es ofrecer servicios multimedia de alta velocidad de datos, baja latencia y fiables en comunicaciones móviles de banda ancha y de baja latencia

Lo descrito anteriormente se logra introduciendo la computación de borde multiacceso (MEC), que integra tecnologías de computación en la nube y de redes inalámbricas. La idea principal de la MEC es asignar recursos informáticos a los usuarios finales dentro de la red de acceso de radio (RAN). El creciente interés por aprovechar los recursos de borde para ofrecer mejores experiencias multimedia ha provocado que surjan múltiples plataformas MEC comerciales. [20]

Las capacidades de almacenamiento de contenido en caché de las redes centradas en la información (ICN) se han combinado con MEC. Esto con el propósito de proporcionar capacidades integradas de almacenamiento en caché,

computación y comunicación (edge-C3). En las aplicaciones multimedia, el edge-C3 es capaz de procesar y almacenar simultáneamente contenido de video en caché para proporcionar a los usuarios servicios de baja latencia y gran ancho de banda. Este mecanismo se muestra en la Figura 7, donde se puede observar que los UE cuentan con capacidades de computación y almacenamiento de mayor potencia. Esto les permite participar también en el edge-C3. Además, el crowdsourcing móvil y la comunicación de dispositivo a dispositivo (*D2D*) permiten que los equipos de los usuarios adyacentes puedan compartir recursos entre sí, lo que reduce la congestión de la red y los recursos que deben utilizarse en los servidores de borde.

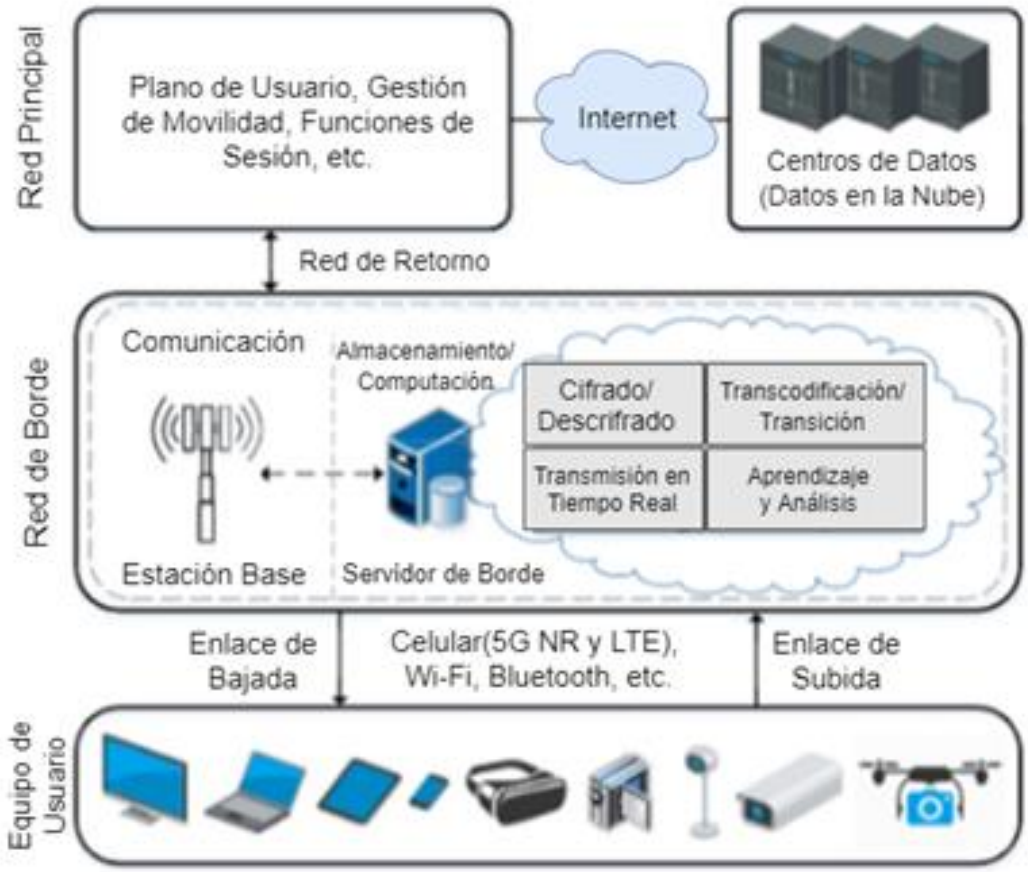


Figura 7. Escenario de video Edge-C3 en redes inalámbricas [20]

3.2 Redes *CDN-P2P*

Las redes *P2P* (*Peer* to *Peer*) son utilizadas para incrementar la capacidad de los sistemas. En comparación con los sistemas convencionales cliente-servidor, las redes *P2P* reducen el tráfico de peticiones a los servidores gracias al trabajo colaborativo realizado por los nodos. En este sentido las redes *P2P* resultan ser

más escalables, es decir, su capacidad aumenta conforme aumenta la cantidad de usuarios conectados al sistema.

En las redes de distribución de contenidos (*CDN*), los servidores son proporcionados por diversos proveedores de servicios de Internet (*ISP*). Estos servidores permiten a los usuarios finales acceder a los diferentes contenidos disponibles en la red, reduciendo el retraso y mejorando la calidad del servicio (QoS); desgraciadamente la cantidad de tráfico de video cada vez es mayor, por lo tanto, las *CDN* se enfrentan con problemas de escalabilidad debido al costo de su implementación.

Por otro lado, como se ha mencionado las redes *P2P* han demostrado ser más escalables pero su rendimiento se degrada por factores asociados al comportamiento de los usuarios, como *freeriding* (*peers* no cooperativos) y *churning* (desconexión inesperada de *peers* habilitados para distribuir el contenido). [5]

En este contexto, los sistemas híbridos *CDN-P2P* muestran ser eficaces por la capacidad de escalabilidad de las redes *P2P* y la estabilidad proporcionada por la infraestructura fija de las *CDN*. Esto se debe a que las redes *CDN-P2P* consideran las características de los archivos de vídeo (la popularidad, la tasa de codificación y el tamaño), las propiedades de las *CDN* (capacidad de carga), las propiedades de las *P2P* (tasas de datos de carga y descarga, el tiempo de permanencia), y los esquemas implementados para asignar los recursos disponibles. [6]

En un inicio las redes *P2P* surgieron como un sistema de distribución de archivos, sin embargo, en años recientes ha incrementado el análisis de este tipo de redes para la distribución de video. En este tipo de servicio es importante resaltar que la reproducción de un video es iniciada a pesar de que su descarga esté en curso.

Las redes *P2P* representan una tecnología estrechamente relacionada con la distribución de video en la quinta generación de sistemas móviles (5G). Las redes *P2P* tienen como base principal al protocolo *BitTorrent* [5], el cual tiene como principal objetivo dividir un archivo de video en diversos segmentos denominados *chunks*. Estos pequeños fragmentos de video permiten que los diversos *peers* en el sistema descarguen un archivo de video intercambiando *chunks* entre sí.

En las redes *P2P*, el protocolo *BitTorrent* separa a los *peers* dentro del sistema en dos grupos: *leechers*, *peers* que contienen una porción del archivo de video. Y *seeders*, *peers* que poseen el archivo de video por completo y permanecen en el sistema con el fin de compartir sus recursos con otros *peers*. Tanto los *seeders* como los *leechers* comparten sus recursos con otros *leechers*.

Cuando un *peer* se une al sistema para descargar el video, se debe contactar con un nodo llamado *tracker*, cuya función principal es crear una lista de los *peers* que poseen una porción o el video en su totalidad. Posteriormente el *tracker* muestra una lista de los *peers* habilitados para compartir sus recursos con el usuario que llegó. Finalmente, el usuario se pone en contacto con los *peers* en la lista y establece que fragmentos de video descargará de cada *peer* con el que esté conectado.

En trabajos como [5] y [6] agrupan un número determinado de *chunks* en segmentos de video relativamente más grandes llamados ventanas. En consecuencia, clasifican a los *peers* de acuerdo con la ventana de video que se encuentran descargando. En la Figura 8 se puede observar la clasificación de los *peers* en un plano cartesiano.

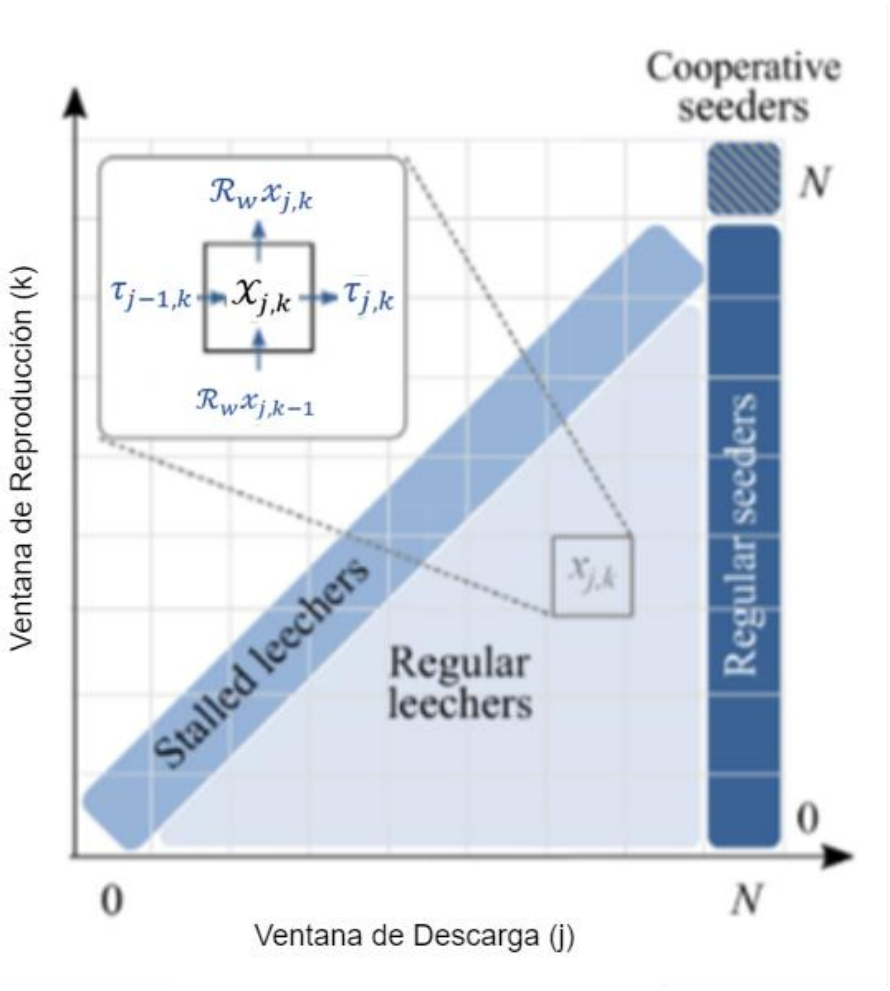


Figura 8. Clasificación de peers en un escenario básico [6]

En los sistemas híbridos *CDN-P2P* dado un esquema de asignación de recursos, se puede obtener la proporción de tasa de subida que proporciona la *CDN* y la proporción de la *P2P* para satisfacer la demanda de un servicio de video. Estas tasas de datos están directamente relacionadas con el costo operativo de la *CDN*

y con la cantidad de recursos que deben proceder de la red *P2P* (energía y el ancho de banda de subida).

3.3 Cadenas de Markov

Generalmente el comportamiento de un sistema, físico o matemático se puede representar describiendo los diferentes estados que puede ocupar e indicando cómo es el cambio entre estos estados. Suponiendo que el sistema que se modela ocupa uno y sólo un estado en cualquier momento del tiempo y su evolución se representa mediante transiciones de estado a estado, se puede decir, que estas transiciones se producen de forma instantánea, es decir, que el paso de un estado a otro no consume tiempo. Si la evolución futura del sistema depende únicamente de su estado actual y no de su historia pasada, el sistema puede representarse mediante un proceso de Markov. Las aplicaciones de los procesos de Markov se pueden encontrar ampliamente en las ciencias biológicas, físicas y sociales, así como en los negocios y la ingeniería (Libro probabilidad).

Un proceso de Markov es un tipo especial de proceso estocástico. Un proceso estocástico se define como una familia de variables aleatorias $\{X(t), t \in T\}$. En otras palabras, cada $X(t)$ es una variable aleatoria y se define en un espacio de probabilidad. El parámetro suele representar el tiempo, por lo que $X(t)$ denota el valor asumido por la variable aleatoria en el momento t. T se denomina conjunto de índices o espacio de parámetros y es un subconjunto de $(-\infty, \infty)$. Si el conjunto de índices es discreto, por ejemplo, $T = \{0,1,2,\dots\}$, entonces tenemos un proceso estocástico de parámetros discretos; de lo contrario, si T es continuo, por ejemplo, $T = \{t: 0 \leq t \leq \infty\}$, llamamos al proceso un proceso estocástico de parámetros continuos.

Los valores que asumen las variables aleatorias $X(t)$ se denominan estados. El conjunto de todos los estados posibles forma el espacio de estados del proceso y éste puede ser discreto o continuo. Si el espacio de estados es discreto, el proceso se denomina cadena y los estados suelen identificarse con el conjunto de números naturales $\{0,1,2,\dots\}$, o un subconjunto de este. Un ejemplo de espacio de estado discreto es el número de clientes de un servicio, mientras que un ejemplo de espacio de estado continuo es el tiempo de espera de un cliente.

Para una cadena de Markov de tiempo discreto, su estado se encuentra definido en un conjunto discreto, pero infinito, de tiempos. Las transiciones de un estado a otro sólo pueden ocurrir, o no ocurrir, en estos instantes de tiempo. Por tanto, se puede representar, sin pérdida de generalidad, el conjunto de índices discretos T del proceso estocástico subyacente con el conjunto de números

naturales $\{0,1,2,\dots\}$. Las observaciones sucesivas definen las variables aleatorias $X_0, X_1, \dots, X_n, \dots$ en los pasos de tiempo $0, 1, \dots, n, \dots$, respectivamente. Formalmente, una cadena de Markov de tiempo discreto $\{X_n, n = 0, 1, 2, \dots\}$ es un proceso estocástico que satisface la siguiente relación, denominada propiedad de Markov [25]:

Para todos los números naturales n y todos los estados x_n

$$Prob\{X_{n+1} = x_{n+1} | X_n = x_n, X_{n-1} = x_{n-1}, \quad X_0 = x_0\}$$

Así, el hecho de que el sistema esté en el estado x_0 en el paso de tiempo 0, en el estado $x_1$en el paso de tiempo 1, y así sucesivamente, hasta el hecho de que esté en el estado x_{n-1}en el paso de tiempo $n-1$ es completamente irrelevante. El estado en el que se encuentra el sistema en el paso de tiempo $n+1$ sólo depende de dónde se encuentre en el paso de tiempo n. El hecho de que la cadena de Markov se encuentre en el estado x_n en el paso de tiempo, n es la suma total de toda la información relativa a la historia de la cadena que es relevante para su evolución futura.

En la cadena de Markov, la abundancia (o penuria) depende del estado del sistema en el tiempo de simulación t.

Puesto que este estado es dinámico, no se puede asegurar que en cualquier instante de tiempo la cadena esté en abundancia o penuria, sino que una u otra condición tienen cierta probabilidad de ocurrir. Si se supone un punto de operación, tal que, según el modelo fluido, garantice abundancia, pero que está relativamente cerca del punto crítico; entonces, durante la evaluación de la cadena de Markov, existe una probabilidad no despreciable de que el sistema caiga en penuria durante algunos intervalos de tiempo, ya que las condiciones de abundancia en el modelo fluido sólo están dadas en términos de los valores promedio X_j. Lo anterior provoca que se reduzca la tasa de transición hacia ventanas superiores durante estos intervalos.

3.4 Servicios de video sobre redes híbridas *CDN-P2P*

El video en vivo a través de internet, incluidos los servicios *OTT* e *IPTV*, han incrementado su polaridad en los últimos años [10]. Estos servicios se basan en el modelo cliente-servidor que utiliza la transmisión adaptativa del protocolo de transmisión de hipertexto (*HTTP*).

Los servicios de video *OTT* comúnmente emplean redes de distribución *CDN* que almacenan el contenido en caché en múltiples ubicaciones geográficas, conocidas como puntos de presencia (*PoP*). *CDN* no solo facilita la carga en el

servidor de origen, sino que también reduce la latencia. Por otro lado, los servicios de *IPTV* emplean la transmisión de multidifusión del protocolo de internet *IP* a través de redes privadas habilitadas para multidifusión, donde una única transmisión es originada en el servidor y se entrega a varios clientes. Sin embargo, el protocolo de multidifusión de *IP* no se admite en la internet abierta, por lo tanto, no está disponible para los servicios de video.

Se han propuesto sistemas híbridos *CDN-P2P* para ofrecer un servicio *CDN* escalable al mismo tiempo que se reduce su ancho de banda y los costos del servidor. Los sistemas más recientes se centran en servicios *CDN-P2P* entregados a navegadores a través de canales de datos *WebRTC* sin depender de complementos de terceros o software de propietarios.

Las principales aplicaciones *CDN-P2P* basadas en *WebRTC* incluyen *SwarmCDN*, *PeerCDN* y *Sharefest* que a su vez condujeron a soluciones comerciales, como *Swarmify, Peer5 y Streamroot*. En estos sistemas, los clientes que solicitan contenido son redirigidos a un servidor centralizado (rastreador) que devuelve una lista de *peers* con el contenido deseado. Sin embargo, el intercambio de contenido entre *peers* no es administrado por el servidor central y estos sistemas pueden no ser compatibles con *NSP*. Un componente importante de los sistemas *P2P* es la formación de grupos.

Existe una serie de diferentes métodos de agrupamiento en la literatura, que se pueden aplicar a la formación de grupos *P2P* basados en mallas.

La aplicación *CDN* es un servicio que se ejecuta en el nodo de cómputo del centro de datos de *NSP Edge*. Extrae contenido de video en vivo desde el servidor de origen al centro de datos *NSP Edge*. Solo un pequeño subconjunto de pares recibe contenido en vivo desde el nodo *CDN*. Todas las demás solicitudes de contenido se redirigen a la aplicación *P2P*.

El control del servicio *P2P-CDN* que se propone en [10] consta de dos fases:

i) La fase de inicio, la aplicación *P2P* envía una serie de fragmentos a los *peers* a través de un árbol de multidifusión para garantizar que cada usuario almacena en búfer un número predeterminado de fragmentos antes de comenzar la reproducción. Se prefiere la multidifusión de *IP* administrada por el controlador *SDN* en la fase de inicio, ya que se sabe que la multidifusión de *IP* tiene un retardo bajo y no se espera un abandono de *peers* durante el corto período de inicio.

ii) En la fase posterior a la reproducción, la aplicación *P2P* realiza una programación de fragmentos pseudo central basada en malla para todos los *peers* en cada grupo *P2P*. En esta etapa se usa la transmisión *P2P* basada en malla porque es más resistente a la

rotación de *peers*. En los sistemas *P2P* de última generación, los *peers* realizan la programación de forma distribuida, lo que conduce a un uso ineficiente de los recursos de la red a medida que los *peers* se favorecen a sí mismos. Se propone una programación pseudo centralizada de fragmentos, realizados en los centros de datos *NSP-edge* para todos los *peers* dentro de cada grupo *P2P,* que sea justa para todos y utilice los recursos de la red de manera más eficiente. El procedimiento es escalable, ya que el número de *peers* en un grupo es limitado.

La fase de inicio continúa hasta que el servidor *CDN* inserta un número predeterminado de fragmentos a lo largo del árbol de multidifusión.

Los *peers* inician la reproducción y entran en la fase posterior a la reproducción tan pronto como almacenan estos fragmentos en búfer. La demora máxima de inicio se calcula cuando todos los *peers* del grupo inician la reproducción. La programación *P2P* basada en malla, posterior a la reproducción, se lleva a cabo en intervalos de tiempo fijos para minimizar la sobrecarga de señalización entre la aplicación *P2P* y los *peers*.

Capítulo 4

Análisis

En este capítulo se abordaron tres tópicos principales que son base teórica para el desarrollo del presente proyecto. El primero es el análisis del modelado de un

sistema de consumo de recursos mediante una cadena de Markov, con la finalidad de conocer las expresiones y la solución por simulación de una cadena de este estilo. El segundo es el análisis de un modelo de servicio de video bajo demanda mediante cadenas de Markov, en este sentido se representa el consumo de recursos en un servicio de *VoD* así como la caracterización del comportamiento de los usuarios. Finalmente, el tercer tema es referente al análisis del funcionamiento de esquemas de asignación de recursos reportados en trabajos relacionados con el objetivo de conocer la forma en que los recursos son asignados dentro de un sistema y los parámetros que influyen en dicha asignación.

4.1 Análisis de la cadena de Markov

En esta sección se realizó el análisis de una cadena de Markov unidimensional que representa a un sistema con pérdidas, arribos markovianos, número finito de servidores y sin cola. Con la finalidad de conocer la evolución en el estado de una cadena de Markov al ocurrir un arribo o pérdida y la forma en que se resuelve una cadena de Markov por simulación.

Dicha cadena representa a un sistema con S servidores disponibles para atender las peticiones de arribo de usuarios que ocurren a tasa λ. En este sistema se conoce el tiempo promedio de duración de un servicio (μ) y por ende se puede calcular la tasa de finalización de un servicio $\left(\frac{j}{\mu}\right)$. La cadena de Markov puede transitar entre el estado 0 y el estado S, es decir, los posibles estados de esta cadena son S+1 estados.

La cadena de Markov se encuentra en estado cero cuando no existen peticiones de arribo de usuarios dentro del sistema y se encuentra en estado S cuando todos los servidores disponibles se encuentran atendiendo a un servicio.

La cadena de Markov analizada considera que existe independencia entre los eventos que pueden ocurrir en el sistema, es decir, en un instante de tiempo ocurre un único evento (petición de arribo o finalización de servicio). Se dice que la cadena es unidimensional debido a que los usuarios son identificados por una sola variable.

Las tasas que provocan una transición en el estado de la cadena son las siguientes:

λ: Representa la velocidad promedio a la que ocurren las peticiones de arribo al sistema y las transiciones del estado i al estado $i + 1$ en la cadena de Markov.

$\left(\frac{j}{\mu}\right)$: Representa la velocidad promedio a la cual finaliza un servicio dentro del sistema. Esta tasa indica la velocidad a la que el estado de la cadena transita del estado i al estado $i-1$ y es el resultado de dividir el estado actual de la cadena j por la duración promedio de un servicio μ.

En la cadena de Markov de la Figura 9 se representa al sistema de pérdidas sin cola, la transición entre los posibles estados y las tasas a las cuales ocurren dichas transiciones en el estado de la cadena.

En la Figura 9 se puede observar que los estados de la cadena se encuentran definidos de 0 a S, donde S es el número de servidores que atienden las peticiones de servicio y también representa el estado máximo de la cadena. De igual forma, en la Figura 9 se observa la transición entre los posibles estados y la tasa a la cual ocurre.

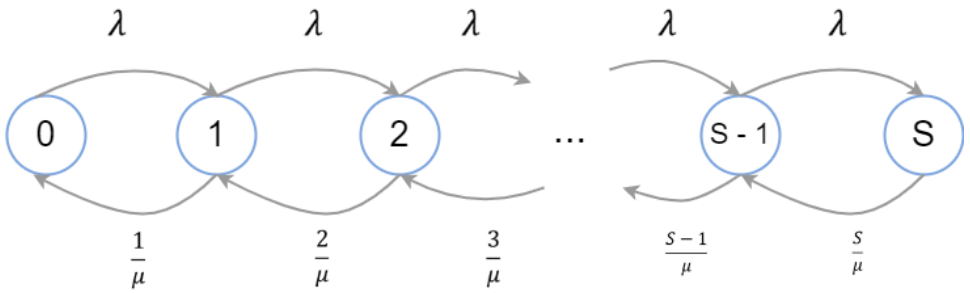


Figura 9. Cadena de Markov Unidimensional para sistema con pérdidas sin cola

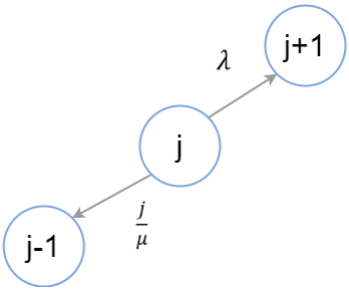


Figura 10. Transición en el estado de la cadena de Markov Unidimensional

En la Figura 10 se representan las posibles transiciones en el estado de la cadena de Markov de un sistema con pérdidas asociadas a su correspondiente tasa de ocurrencia, es decir, la cadena transita al estado superior $(j+1)$ a tasa λ o transita al estado inferior $(j-1)$ a tasa $\frac{j}{\mu}$.

Las transiciones de estado son eventos discretos, es decir, en un instante de tiempo sólo puede ocurrir una transición al estado superior o una transición al estado inferior. Estos eventos se implementan generando una variable aleatoria

con distribución exponencial negativa con su respectiva tasa y se hace un acumulado del tiempo de estancia por estado para de esta forma obtener estadísticas de consumo.

Una vez analizada la cadena de Markov que modela un sistema con pérdidas y sin cola se realizó un diagrama de flujo ilustrado en la Figura 11, para dar solución por medio de simulación a la cadena de Markov y finalmente se implementó esta simulación en el IDE Matlab para obtener estadísticas de consumo y desempeño del sistema.

La implementación inicia definiendo los parámetros de entrada, es decir, las tasas que modificarán la posición del usuario en el estado de la cadena. Las variables más notables son: tasa de arribo de usuarios (λ), duración promedio de un servicio (μ) y número de servidores (S).

Posteriormente, se definen los siguientes vectores de longitud $S + 1$: tiempos de estancia por estado $(tiempo)$ y arribos por estado $(contador)$.

En el caso de que la cadena se encuentre en estado cero únicamente se genera una variable aleatoria con distribución exponencial negativa a partir de la tasa de arribo de usuarios (λ) y se acumula su valor en tiempo de estancia en la posición 1 y se incrementa en una unidad al contador de arribos del estado 0.

En caso de que la cadena se encuentre en otro estado, contenido en $[1, S]$, se generan 2 variables exponenciales negativas (A y B) a partir de la tasa de arribo de usuarios (λ) y la duración promedio de un servicio (μ) respectivamente. Posteriormente se elegirá la de menor valor para determinar qué evento ocurrió (petición de arribo o finalización de servicio) y se tienen los siguientes 2 casos:

* Petición de arribo: Se almacena el valor de A en tiempo de estancia por estado en la posición $(EA: estado\ anterior)$ y el contador de arribos por estado se incrementa en una unidad en la posición $(EA + 1)$.

* Finalización de servicio: Se almacena el valor de B en tiempo de estancia por estado en la posición (EA).

La implementación de la solución matemática se realizó conforme al diagrama de flujo de la Figura 11.

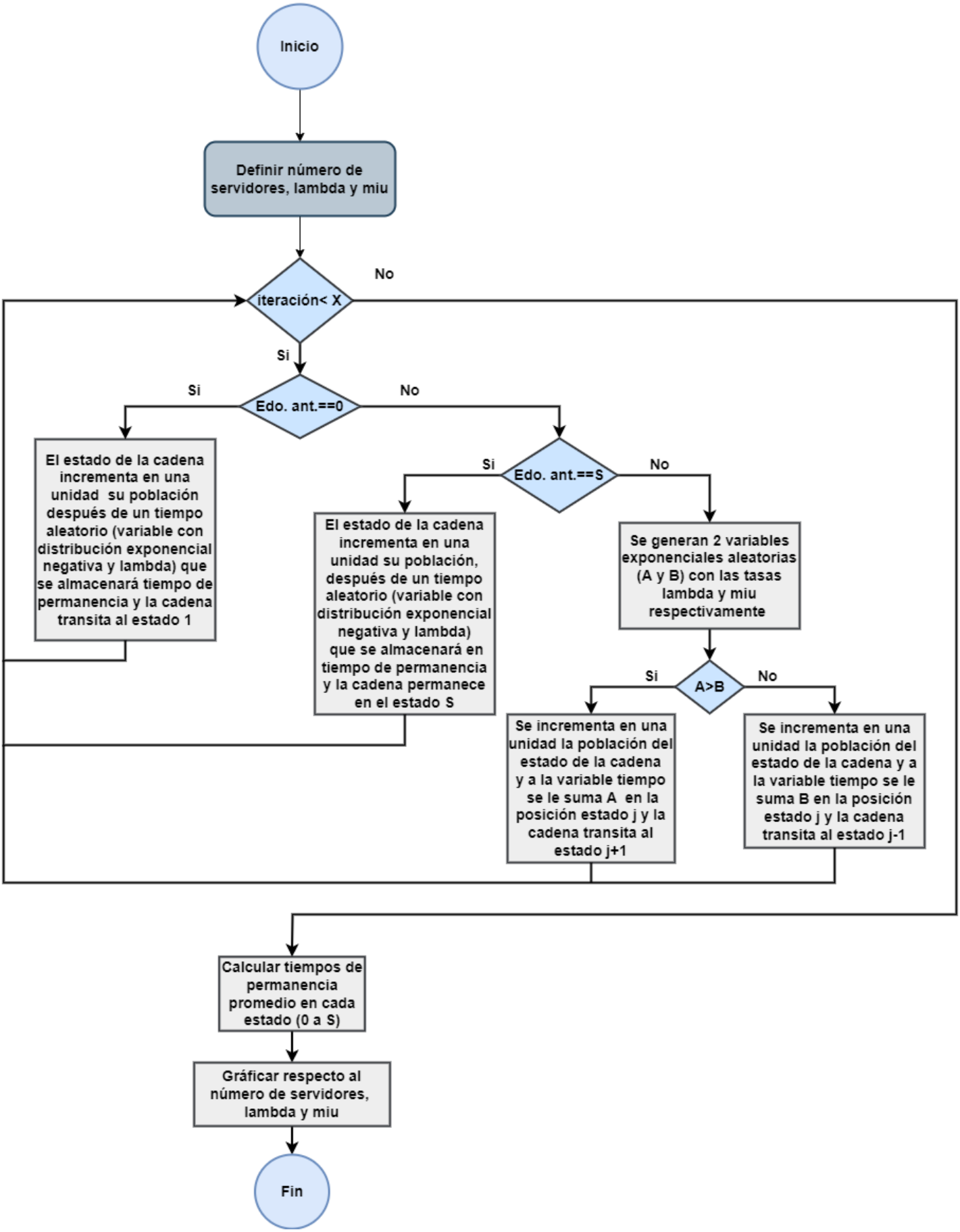


Figura 11. Diagrama de flujo para dar solución por simulación a la cadena de Markov

La cadena de Markov que representa un sistema con pérdidas sin cola se solucionó por dos métodos: el analítico y el numérico.

La solución analítica se realizó mediante la expresión Erlang-B, que permite conocer el consumo de recursos dentro de un sistema tomando en cuenta que este consumo depende del producto de la tasa de arribo de usuarios (λ) y la duración promedio de un servicio(μ).

$$P_j = \frac{\frac{(\lambda\mu)^j}{j!}}{\sum_{k=0}^{S} \frac{(\lambda\mu)^k}{k!}}$$

Ecuación Erlang-B

La solución numérica se llevó a cabo mediante dos técnicas: tiempo promedio de estancia por estado y cantidad de arribos por estado.

- Tiempo promedio por estado: esta técnica consiste en generar dos variables aleatorias con distribución exponencial negativa (A y B). A se genera con la tasa promedio de arribos de usuarios (λ) y B se genera con la tasa promedio de finalización de servicio $\left(\frac{j}{\mu}\right)$.

  Una vez que se obtienen estas dos variables se elige la de menor valor. En caso de que la variable A haya sido el mínimo se dice que ocurrió un arribo en el sistema. En caso contrario, cuando la variable de menor valor es B se entiende que un servicio que estaba en curso ha terminado.

  Cuando no existen peticiones de arribo, el estado de la cadena se encuentra en 0. El estado de la cadena incrementa en una unidad de manera uniforme hasta S a medida que se genera una petición de arribo. 0 es el estado mínimo de la cadena de Markov y cuando esta se encuentra en ese estado solo puede ocurrir una petición de arribo y por ende se genera únicamente una variable aleatoria con distribución exponencial negativa y tasa de arribos (λ).

  S es el estado máximo al cual puede transitar la cadena de Markov, por lo tanto, cuando la cadena se encuentra en este estado únicamente se debe generar una variable aleatoria con distribución exponencial negativa y tasa de finalización de servicio $\left(\frac{j}{\mu}\right)$.

  Al momento de conocer qué evento fue el que sucedió, el estado de la cadena transita a otro estado. En caso de que haya ocurrido una petición de arribos, la cadena transitará del estado i al estado $i+1$. En caso contrario si lo que ocurrió fue la finalización de un servicio, la cadena transitará del estado i al estado $i-1$.

Entonces el tiempo promedio por estado se incrementará en la posición i en un valor A o B si lo que ocurrió fue una petición de arribos o la finalización de un servicio, respectivamente. Y finalmente se obtienen el tiempo de estancia promedio final por estado, el cual que indica el consumo de recursos por estado del sistema y se grafica en función de las variables de entrada.

- Arribos por estado: esta técnica consiste en generar dos variables aleatorias con distribución exponencial negativa (A y B). A se genera con la tasa promedio de arribos de usuarios (λ) y B se genera con la tasa promedio de finalización de servicio $\left(\frac{j}{\mu}\right)$.

  Al igual que la técnica de tiempo promedio por estado se elige la de menor valor con la finalidad de conocer si ocurrió una petición de arribo al sistema (A) o la finalización de un servicio (B).

  La transición y definición del estado de la cadena de Markov es la misma que en la técnica anterior. Por lo tanto, una vez que se conoce qué evento sucedió, el estado de la cadena transita a otro estado. En caso de que haya ocurrido una petición de arribos, la cadena transitará del estado i al estado $i+1$. En caso contrario si lo que ocurrió fue la finalización de un servicio, la cadena transitará del estado i al estado $i-1$.

  Entonces los arribos por estado se incrementarán en una unidad en la posición $i+1$ en caso de que la variable A haya sido el mínimo (petición de arribo) o se incrementarán en una unidad en la posición $i-1$ si lo que ocurrió fue la finalización de un servicio (B). Y finalmente se obtienen los arribos finales por estado.

  Estos arribos permiten conocer en qué medida se realizó el consumo de recursos por estado del sistema y son graficados en función de las variables de entrada.

Una vez obtenidos los resultados de la implementación se comparó gráficamente la solución analítica contra las soluciones numéricas (tiempo promedio por estado y arribos por estado) como se muestra en las Figuras 12-14.

En dichas gráficas se observa que el consumo de recursos muestra una curva similar, sin embargo, el valor promedio de consumo varía de acuerdo con el producto de la tasa de arribos y la duración promedio de servicio.

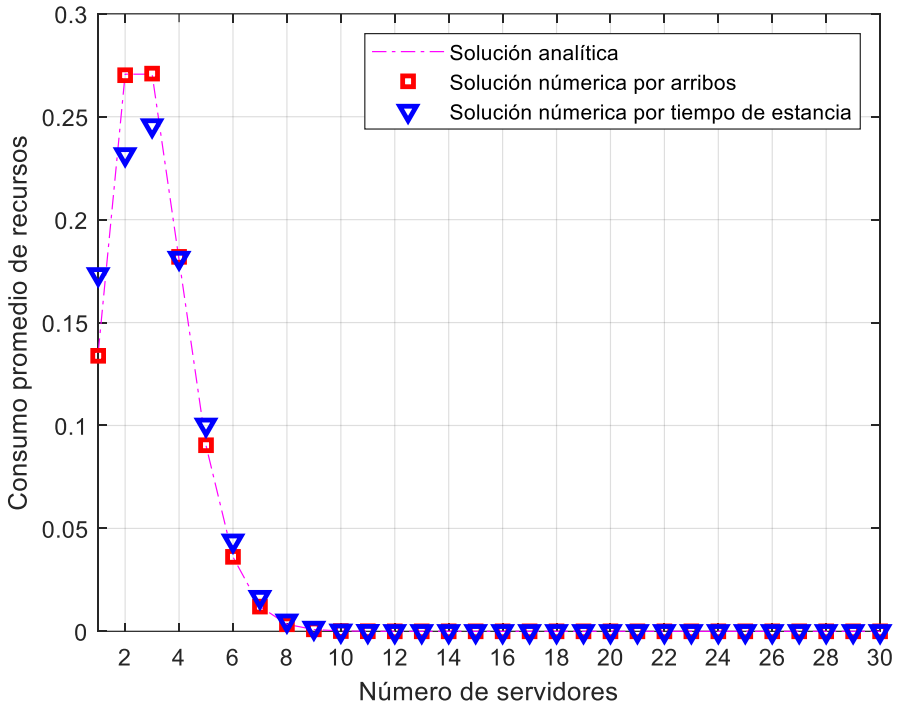


Figura 12. Gráfica de solución por implementación de la cadena de Markov para S=30, $\lambda = 4$ y $\mu = 0.5$

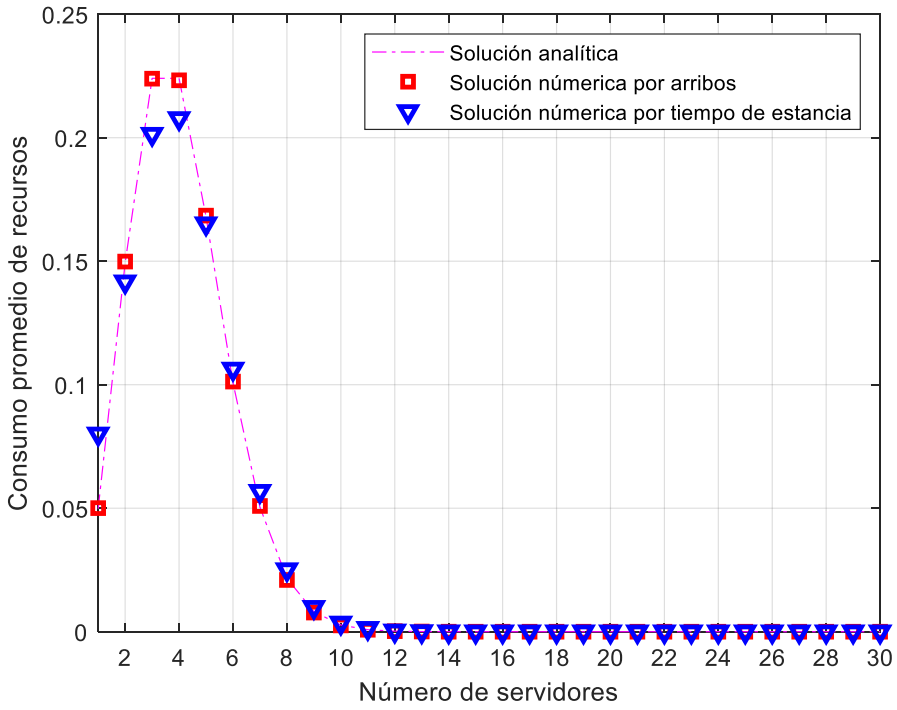


Figura 13. Gráfica de solución por implementación de la cadena de Markov para S=30, $\lambda = 6$ y $\mu = 0.5$

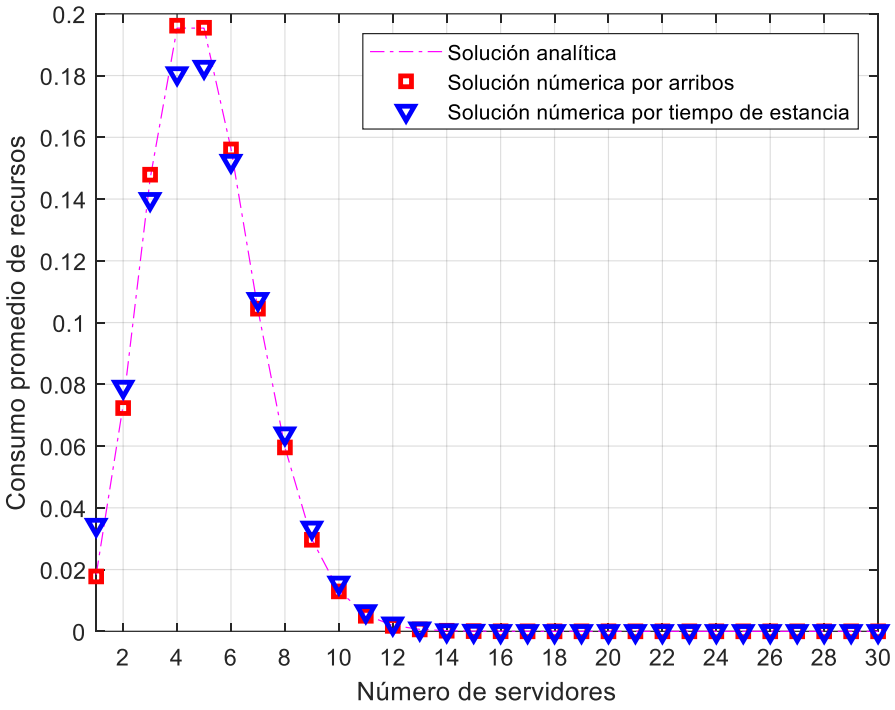


Figura 14. Gráfica de solución por implementación de la cadena de Markov para S=30, $\lambda = 8$ y $\mu = 0.5$

4.2 Análisis de modelos para servicios de video bajo demanda

El video bajo de manda *VoD (Video on Demand)* es un archivo multimedia que puede surgir directamente de un video en vivo, en el caso que el evento se guarde una vez finalizado o bien desde el principio archivo multimedia puede ser diseñado, editado y almacenado para catalogarlo como *VoD*.

Este archivo es previamente almacenado dentro de un servidor de tal manera que los usuarios puedan acceder a el mediante internet en el momento que lo requieran. En los servicios de video bajo demanda el usuario tiene la capacidad de elegir el video de su preferencia. En comparación con los servicios de video en vivo, el video bajo demanda puede ser manipulado por el usuario, es decir, el usuario puede reproducir, adelantar, retrasar o pausar el video cuando el usuario así lo requiera, estas acciones no las puede realizar un usuario cuando consume un servicio de video en vivo.

La naturaleza de un archivo de video implica que este sea producido en pequeños segmentos o *frames* llamados *chunks* de corta duración. Debido a que en el caso de video bajo demanda (*VoD*), la duración total del video es finita y por lo tanto se conoce el número de *chunks* que componen al video, en trabajos

como [12], se opta por agrupar una catidad finita de *chunks* (n) en un segmento de video llamado ventana.

Las N ventanas que componen al video son de igual tamaño. Es decir, un archivo de video bajo demanda, está compuesto por N ventanas y estás a su vez están compuestas por n *chunks,* que siguen el flujo de la estructura del *VoD* como se observa en la Figura 15.

Figura 15. Estructura de un archivo de video bajo demanda-VoD

La agrupación de *chunks* en ventanas se realiza con la finalidad de clasificar a los *peers* dentro del sistema de acuerdo con el número de ventana que están descargando. Por otro lado, esta agruparción permite distribuir de forma eficiente el contenido entre todos los *peers* que solicitan el *VoD* dentro de un sistema *P2P.*

Basado en un modelo fluido para la evaluación de los servicios del *VoD* en el cual se consideran las características del servicio como la tasa de codificación, la popularidad del video, los atributos de la red, la capacidad de tasa de carga de datos de los servidores y *peers* que son los consumidores del servicio. Se tiene la siguiente clasificación de usuarios que consumen un servicio de video bajo demanda:

- *Peers*: son aquellos usuarios interesados en descargar un mismo archivo de video.

- Seeds: son *peers* que poseen en su totalidad el archivo de video almacenado en su *buffer*.

- Downloaders: son *peers* que únicamente poseen una fracción del archivo de video completo.

De manera general tanto *seed*s como *downloaders* son *peers* conectados al sistema. Un *seed* es un downloader que ha finalizado el proceso de descarga del archivo, pero permanece en el sistema para compartir sus recursos con otros *peers*. En la Figura 16 se observa la distribución y posición de estos usuarios dentro de la red.

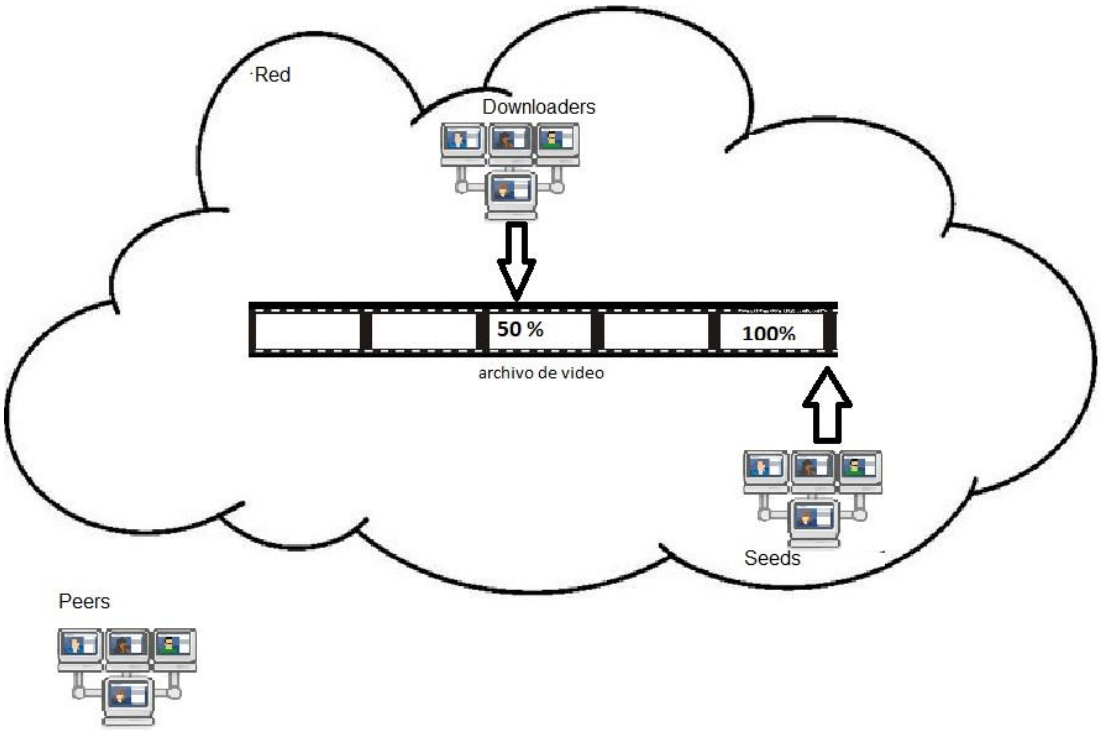


Figura 16. Clasificación de usuarios dentro de una red P2P

Dentro de un sistema híbrido *P2P-CDN* los recursos son proporcionados por los *peers* conectados al sistema y por los servidores *CDN*. Por lo tanto, los *peers* que consumen recursos para descargar el contenido en este tipo de sistemas también pueden atender a otros *peers* que necesiten recursos para continuar con la descarga del video. Un *peer* tiene la capacidad de compartir sus recursos con *peers* que presentan un menor progreso en la descarga del video, es decir, un *peer* que se encuentra descargando la ventana i, tiene almacenadas en su *buffer* las ventanas $[0, i-1]$.

Lo anterior indica entonces, que este *peer* puede proporcionar estos recursos a *peers* que deseen descargar las ventanas $[0, i-1]$. Este *peer* no puede atender a *peers* descargando vetanas superiores a la ventana i puesto que aún no cuenta con estos recursos. Esta forma de asignar recursos se conoce como distribución uniforme y no se asignan únicamente los recursos de un *peer* en particular sino que se asignan los recursos proporcionados por la población de *peers* que se encuentran descargando una ventana en especifico como se observa en la Figura 17.

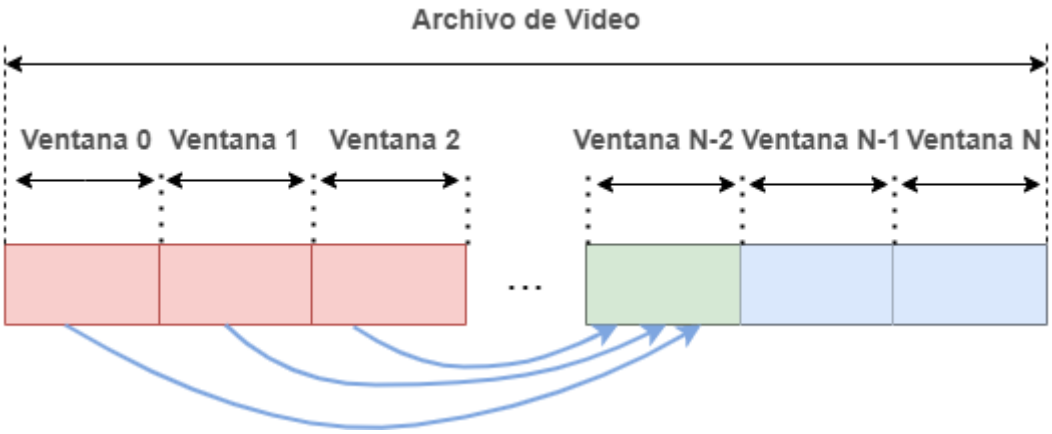


Figura 17. Distribución de recursos en una red P2P

En la Figura 17 se puede observar cómo es que las poblaciones de *peers* de las ventanas $[0, N-3]$ son atendidas por la población en la ventana $N-2$. De forma análoga estas poblaciones también reciben recursos proporcionados por las poblaciones $N-1$ y N.

El esquema de asignación de recursos con distribución uniforme asigna los recursos considerando la posición de una población de *peers* y el tamaño de dicha población.

Dentro de la distribución de recursos para un sistema de consumo de servicio de video bajo demanda, se tienen dos condiciones en las cuales se puede encontrar el sistema, abundancia y penuria.

En el caso de que los recursos de subida proporcionados por la población de una ventana sean mayores a los recursos de descarga requeridos por las poblaciones en ventanas inferiores se dice que el sistema se encuentra en condición de abundancia.

En el caso contrario, cuando los recursos de subida son menores a los recursos de descarga requeridos, es decir, el sistema se encuentra en condición de penuria y debe solicitar recursos a la red *CDN.*

En la Figura 18 se observa la cadena de Markov asociada a un sistema de servicio de video bajo demanda (*VoD*). Mediante esa cadena de Markov se representa el comportamiento que tienen los usuarios una vez que se encuentran conectados al sistema y consumiendo el contenido. De manera general, en los servicios de video bajo demanda se identifican tres principales sucesos que pueden ocasionar una transición en el estado de la cadena de Markov.

El estado de la cadena es representado con un vector de longitud N, que contiene las poblaciones que están descargando cada una de las ventanas que componen al archivo de video.

La transición en el estado de la cadena se puede generar una vez que ocurra el arribo de un usuario al sistema, el abandono del sistema por parte de un usuario o la transferencia de un usuario a la ventana superior inmediata.

- El arribo de un usuario al sistema es medido a través de la tasa λ, la cual indica la velocidad promedio a la cual nuevos usuarios arriban al sistema. El arribo de un nuevo usuario es generado en la ventana 0, debido a que de manera natural un usuario comienza a visualizar el video desde el inicio.

- El abandono de un usuario al sistema tiene dos vertientes: un usuario que se encuentra descargando una ventana en el intervalo $[0, N-1]$ abandona el sistema a tasa θ, que indica la velocidad promedio a la cual un usuario abandona el sistema, este abandono se puede generar por diversas circunstancias.

  La otra vertiente es cuando el usuario pertenece a la ventana N, estos usuarios abandonarán el sistema a una tasa promedio γ. Lo anterior, se debe a que estos usuarios han completado la descarga del video en su totalidad y permanecen en el sistema para atender a otros usuarios que necesiten recursos para continuar su descarga. De igual forma, estos usuarios pueden permanecer en el sistema para terminar de visualizar el video y por lo tanto son fuente de recursos.

- La transferencia de un usuario a la ventana superior inmediata se genera cuando un usuario ha terminado de descargar la ventana i y desea comenzar a descargar la ventana inmediata superior $(i+1)$. Esta transferencia ocurre a tasa τ_i, que indica la velocidad promedio a la cual un usuario es transferido a la ventana superior inmediata.

  La transferencia de un usuario no existe para la población en la ventana N, debido a que esta ventana es la última que componen al archivo de video y por lo tanto no existe otra ventana que descargar.

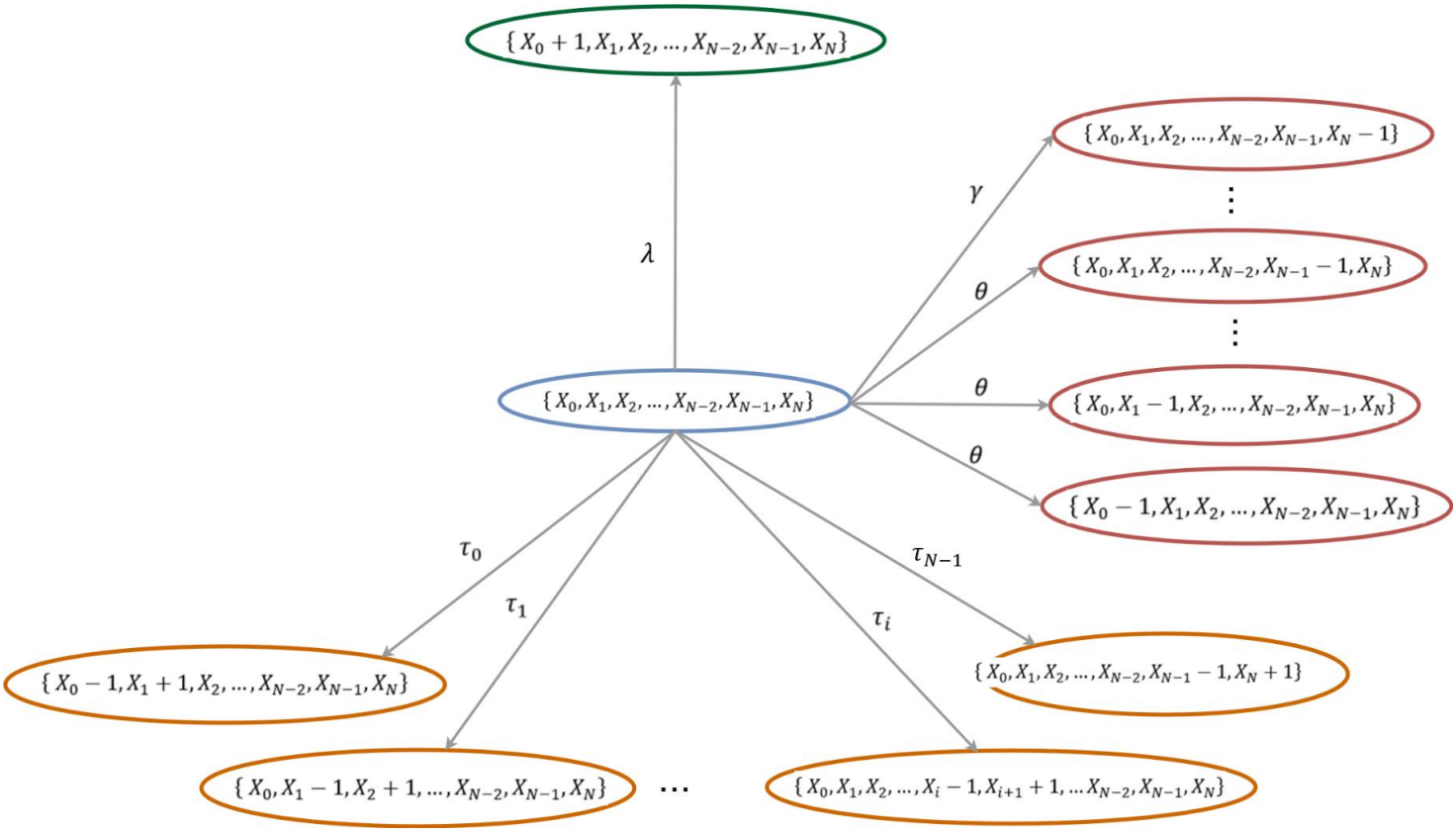


Figura 18. Cadena de Markov para un sistema de consumo de servicio de video bajo demanda (VoD)

4.3 Análisis esquemas de asignación de recursos para servicios de video bajo demanda

En esta sección se realizó el análisis de los esquemas de asignación de recursos reportados en los trabajos de investigación relacionados a este proyecto. En primer lugar, se describe a cada uno ellos con la finalidad de conocer sus principales características y funcionamiento para poder seleccionar algunos de los esquemas descritos, que pueden adecuarse para los servicios de video en vivo, y que serán tomados como base para realizar el bosquejo del esquema de asignación de recursos a proponer en el en el presente proyecto.

En el siguiente listado se reportan los esquemas de asignación de recursos para servicios de video bajo demanda (*VoD*) que presentan mayor relación con el presente proyecto.

- Esquema de Distribución Uniforme (DU): este esquema consiste en distribuir los recursos disponibles de una ventana de manera uniforme entre todos los usuarios que han arribado al sistema, es decir, los recursos son asignados de igual manera entre los diversos usuarios que están consumiendo el archivo de video.

En este esquema el parámetro para asignar los recursos ofrecidos por *seeds, downloaders y* servidores *CDN* es que los usuarios pertenecientes a las ventanas 0 a $N-1$ son atendidos por usuarios que presentan un mayor progreso en su descarga. Por lo tanto, los usuarios situados en ventanas i son atendidos por usuarios en ventanas $j, j > i.$

De manera general a los usuarios situados en las ventanas 0 a $N-1$ reciben el nombre de *downloaders* y su principal característica es que pueden descargar el archivo de video y compartir el progreso de descarga que tienen almacenado en *buffer* con los demás usuarios.

En este esquema los usuarios situados en la ventana N son llamados *seeds* cuya principal característica es que cuentan con la totalidad del archivo de video almacenado en *buffer* y por ende representan una fuente de recursos.

- Esquema Q ventanas hacia atrás: este esquema al igual que el esquema DU consiste en distribuir los recursos disponibles de una ventana de manera uniforme entre todos los usuarios posicionados en ventanas inferiores a las que se atiende.

  La principal diferencia entre Q ventanas hacia atrás y DU es la definición de un parámetro de control llamado Q. Dicho parámetro limita las ventanas inferiores a las cuales se atenderá, ya que Q representa el número de ventanas inferiores que tendrán acceso a los recursos de una ventana.

  El objetivo principal de este parámetro es atender con mayor prioridad a las ventanas inmediatas inferiores y generar de esta forma una asignación de recursos más equitativa.

  En este esquema al igual que en DU los *downloaders* son atendidos por los servidores *CDN, seeds y* con recursos de otros *downloaders*.

- Esquema de Generalización de Distribución por Priorización de Ventanas (GDPV): este esquema al igual que DU, consiste en asignar los recursos disponibles de una ventana entre los *downloaders* posicionados en ventanas inferiores.

  A diferencia de DU, en este esquema se establece un parámetro de prioridad denotado por ε. Este parámetro define el grado de prioridad con el que se atenderá a *downloaders* situados en ventanas inferiores a la ventana de donde se obtienen los recursos.
  ε define una alta prioridad a ventanas altas, es decir, a ventanas i con mayor cercanía a la ventana N. Esto con el fin de evitar condiciones de penuria en el sistema. El parámetro de priorización genera que una

ventana atienda con mayor prioridad a ventanas inmediatas inferiores y así propiciar una mayor colaboración entre todos los *downloaders* conectados al sistema.

En este esquema los recursos ofrecidos por *seeds* y servidores *CDN* son aginados a *downloaders* en ventanas superiores con mayor prioridad.

- Esquema *Inmediate-Neighbors Uniform Allocation (INUA)*: este esquema establece que un *downloader* puede seleccionar aleatoriamente los recursos de *peers* con mayor progreso en su descarga, es decir, un *downloader* puede seleccionar a los *peers* en ventanas superiores que le proporcionarán recursos.

  Sin embargo, la selección se debe realizar bajo la condición $M = QX$. Esta condición limita el número de *peers* en ventanas hacia adelante que pueden atender a *downloaders* en ventanas inferiores. En dicha condición M representa los *peers* seleccionados para proporcionar recursos, Q el número de ventanas hacia adelante y X la cantidad de *peers* en las ventanas superiores.

- Esquema *Prioritazed Window Distribution (PWD)*: este esquema consiste al igual que GDPV consiste en asignar los recursos provenientes de *seeds* y servidores *CDN* con mayor prioridad *downloaders* que se encuentran descargando ventanas superiores.

  De igual manera en este esquema se emplea el parámetro ε para definir la prioridad con que serán asignados los recursos de una ventana a ventanas inferiores. Esto con la finalidad de que los recursos se distribuyan de una manera más equitativa y el sistema en general se encuentre en condiciones de abundancia.

  La principal diferencia que existe entre *PWD* y GDPV, es que en *PWD* únicamente los *seeds* y servidores *CDN* pueden proporcionar recursos a ventanas inferiores mientras que en GDPV se considera también a los recursos provenientes de los *downloaders*.

La Tabla 3 muestra una comparación entre los esquemas de asignación de recursos analizados previamente.

Esquema	Parámetros	Ventajas	Desventajas
Distribución uniforme (DU)	Se basa en la posición de la ventana y el tamaño de su población. Los *peers* son atendidos por *peers* en ventanas j>i.	Los recursos se distribuyen de forma homogénea en proporción al tamaño de la población en una ventana.	Todas las ventanas desde 0 a j-1 se consideran inferiores y por lo tanto se deben atender con la misma prioridad.
Q ventanas hacia atrás	Q número de ventanas inferiores	Los recursos se distribuyen de forma homogénea de acuerdo con el tamaño de la población en la ventana y el limitando el número de ventanas inferiores.	Al ser Q un parámetro estático se debe definir bien los límites de los bordes del video (ventanas 0 y N). Existe la probabilidad de caer en condiciones de penuria si dentro del rango de ventanas no hay disponibles suficientes recursos.
GDPV	ε grado de prioridad	Organiza las ventanas en orden de prioridades. Atiende primero a las ventanas con la prioridad más alta.	Dificultad de implementación por la función de prioridad que se utiliza. Existe posibilidad de generar condición de penuria.
INUA	$M = QX$	Un usuario puede seleccionar quien le proporcione recursos.	No hay homogeneidad al momento de compartir los recursos.

PWD	ε grado de prioridad	Organización por orden de prioridad a las ventanas	Únicamente los *seeds* y servidores *CDN* pueden compartir recursos.

Tabla 3. Características principales de esquemas de asignación de recursos investigados

Como resultado de analizar cada uno de los esquemas de asignación de recursos reportados en la Tabla 3 se decidió tomar como referencia a los esquemas:

- Esquema de asignación de recursos de distribución uniforme (DU)
- Esquema Q ventanas hacia atrás
- Esquema GDPV

Esto gracias a que, a pesar de que son esquemas para servicios de video bajo demanda (*VoD*), tienen características similares que se pueden adaptar y mejorar para servicios de video en vivo.

Con base en el análisis previo de los esquemas de asignación de recursos seleccionados se realizó un seudocódigo que permite comprender el funcionamiento de cada uno de estos.

De forma general se identificó que todos los esquemas presentan el mismo comportamiento una vez que se ha determinado cuales son los *peers* de donde se obtendrán los recursos que utiliza un *downloader* para descargar el archivo de video y por lo tanto son similares a partir de esta condición.

Esquema de Asignación de Recursos Uniforme

Inicio

1.- Llega un usuario nuevo al sistema y se posiciona en la población que está descargando la ventana i
2.- Se define $k = i + 1$
3.- Repetición de Recolección
Se recolectan todos los *seeds* y *downloaders* de las poblaciones que se encuentren con las ventanas $i + 1 \leq k \leq N$
Fin de Repetición
4.- Listado de *peers* con recursos disponibles dentro de la población correspondiente
5.- ¿La tasa real de descarga es igual a la ideal de descarga?

Si: Almacenar los *peers* que asignan recursos y la cantidad de recursos

No: Se asignan recursos, ¿aún hay *peers* en la lista?
Si: Retornar a paso 4
No: Buscar recursos en el *CDN* y repetir condición Si del paso 5
6.- Generar estadísticas

Fin

Esquema Q ventanas hacia atrás

Inicio

1.- Llega una población nueva de usuarios al sistema en la ventana i
2.- Se define Q que es el límite de ventanas hacia atrás
3.- Repetición de Recolección
Se recolectan todos los *peers* de las poblaciones en ventanas $i+1 \leq k \leq i+Q$
Fin de Repetición
4.- Listado las poblaciones con recursos disponibles
5.- ¿La tasa real de descarga es igual a la ideal de descarga?

Si: Almacenar las poblaciones de *peers* que asignan recursos y la cantidad de recursos

No: Se asignan recursos, ¿aún hay poblaciones nuevas en la lista?
Si: Retornar a paso 4
No: Buscar recursos en el *CDN* y repetir condición Si del paso 5
6.- Generar estadísticas

Fin

Esquema DVPG

Inicio

1.- Llega un usuario nuevo al sistema a la ventana i
2.- Se define ε que es el parámetro de control
3.- Repetición de Recolección
Se recolectan todos *peers* que presenten con un índice $(j+1)^{\varepsilon}$ alto
Fin de Repetición
4.- Listado de *peers* con recursos disponibles
5.- ¿La tasa real de descarga es igual a la ideal de descarga?

Si: Almacenar los *peers* que asignan recursos y la cantidad de recursos

No: Se asignan recursos, ¿aún hay *peers* en la lista?
Si: Retornar a paso 4
No: Buscar recursos en el *CDN* y repetir condición Si del paso 5
6.- Generar estadísticas

Fin

Aunado a esto se analizaron las expresiones para asignar recursos de los esquemas Q ventanas hacia atrás y DVPG para encontrar las diferencias para servicios de video bajo demanda y servicios de video en vivo.

Esquema Q ventanas hacia atrás

La siguiente expresión determina la tasa promedio necesaria para descargar una ventana del video:

$$c_\omega \leq \mu_\omega M \left(\sum_{J=j+1}^{mín(j+q,N-1)} \frac{X_{JM}}{\sum_{k=\max(0,J-q)}^{J-1} x_{kM}+x_{kF}} + \frac{Y_M}{\sum_{k=N-q-1}^{N-1} x_{kM}+x_{kF}} \right) +$$
$$\mu_\omega F \left(\sum_{J=j+1}^{mín(j+q,N-1)} \frac{X_{JF}}{\sum_{k=\max(0,J-q)}^{J-1} x_{kM}+x_{kF}} + \frac{Y_F}{\sum_{k=N-q-1}^{N-1} x_{kM}+x_{kF}} \right) + \frac{v}{X}$$

En primer lugar, el esquema de asignación de recursos a proponer en el presente proyecto no considera 2 regiones o zonas donde están distribuidos los *peers* dentro del sistema, por lo tanto, no se tomará a $M\ y\ F$ como variables diferentes.

Debido a que no existen usuarios del tipo *seed* el esquema prescindirá del término Y, que representa los recursos provenientes de *peers* conectados a la ventana c.

El término $\frac{v}{X}$ para el desarrollo de este proyecto va a representar los recursos que proporciona la red *CDN.*

Finalmente, considerando que el número de ventanas queda restringido por el parámetro c, los límites en la suma de los recursos cambiarán de forma significativa y el parámetro Q (número de ventanas hacia atrás) deberá ser $0 < Q \leq c$.

Esquema GDPV

La siguiente expresión determina la tasa promedio necesaria para descargar una ventana del video:

$$c_\omega \leq (j+1)^\varepsilon \left(\sum_{J=j+1}^{N} \frac{\mu_\omega X_J}{V_{J-1}^*} + \frac{v}{V_{N-1}^*} \right), para\ 0 \leq j \leq N-1$$

La expresión anterior se puede adecuar de manera correcta a los servicios de video en vivo debido a que considera tanto recursos de la red *P2P* como de la red *CDN*, además de que considera a todos los *peers* dentro del sistema para que reciban recursos de manera equitativa o correspondiente a su progreso de descarga.

Sin embargo, se debe limitar la expresión anterior al caso de los videos en vivo. Debido a que en una transmisión de video en directo no existen usuarios del tipo

seed los recursos dentro la red provienen exclusivamente de *peers* conectados a ventanas $i, 0 \leq i \leq c$ y de los servidores *CDN.*

Por lo mencionado anteriormente el límite de la suma no llegará hasta N sino que se cambiará esta variable por una c, que identifica a la ventana actual *(currently)*. A su vez en el presente proyecto se busca que los usuarios más cercanos a la ventana c sean los que reciban con mayor prioridad los recursos provenientes de los servidores *CDN.* Por lo tanto, el término $\frac{v}{V_{N-1}^{*}}$, seguramente será restringido a poblaciones cercanas a la ventana c.

El valor de la variable j siempre debe corresponder al índice de la ventana a donde pertenece la población que va a recibir los recursos y por lo tanto se debe limitar a $0 \leq j \leq c - 1$.

Finalmente, el parámetro de priorización ε para el caso de este proyecto puede ser cualquier número real e indicará la prioridad con que una población de *peers* puede acceder a los recursos ofrecidos por las poblaciones en ventanas superiores.

Como se había mencionado anteriormente el esquema de asignación de recursos que se propondrá en este proyecto puede ser la fusión de los 2 esquemas mencionados anteriormente, con los cual quizá vuelvan a cambiar los límites de la suma.

Capítulo 5

Diseño

En este capítulo se describe el diseño y estructura de la cadena de Markov que representa a los servicios de video en vivo, así como el diseño del esquema de asignación de recursos que se ha propuesto para este tipo de servicios.

5.1 Diseño de la cadena de Markov para servicios de video en vivo

En esta sección se describe la cadena de Markov que representa el comportamiento de los usuarios una vez que se conectan a un sistema de servicio de video en vivo, de igual forma se explican los posibles estados que puede tomar la cadena dado un evento que modifique su estado actual.

En este proyecto se supone la distribución de un archivo de video en vivo sobre una red híbrida *P2P-CDN*. Como se ha mencionado anteriormente, los videos son generados por segmentos pequeños llamados *frames*, en este ámbito los denominaremos *chunks*, que es la unidad indivisible de un video.

Los *peers* conectados al sistema descargan el archivo de video *chunk* a *chunk*, sin embargo, con el objeto de no tener una gran cantidad de poblaciones de *peers* descargando el archivo de video, se agrupan N *chunks* en segmentos de video más grandes llamados ventanas. El tamaño de las ventanas influye en la manera en que se distribuyen los recursos entre *peers*, pues al haber ventanas más grandes, la distribución se realiza entre un menor número de grupos de *peers* (poblaciones). Se espera que cada uno de estos grupos sea de un tamaño considerable para una distribución eficiente. Lo anterior se representa en la Figura 19.

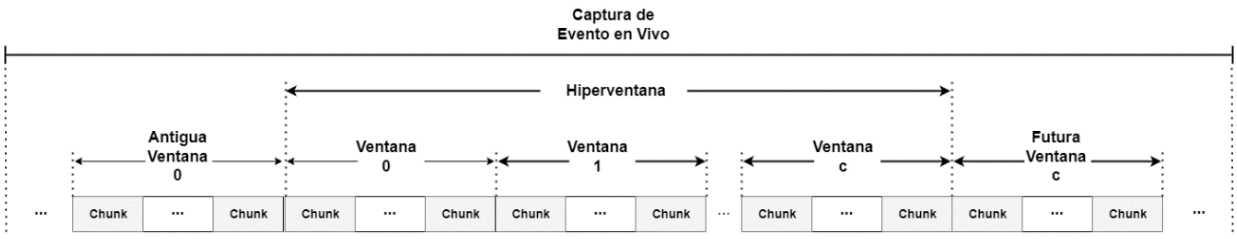


Figura 19. Estructura de una hiperventana

En el caso de video bajo demanda (*VoD*) se conoce la duración total del archivo de video y por lo tanto se divide en N ventanas de igual tamaño (n *chunks*) al momento de distribuirlo entre los usuarios. Sin embargo, para el caso de video en vivo no se conoce la duración total del archivo de video puesto que no se sabe con exactitud el tiempo que durará la captura del evento en tiempo real.

Considerando lo anterior, se optó por definir un contenedor llamado hiperventana, este contenedor es de longitud c, es decir, la hiperventana contiene de la ventana 0 a la ventana c y almacena las ventanas en tiempo real. La ventana 0, es la última ventana que se considera para que el usuario visualice el evento en tiempo real, por otro lado, la ventana c representa el fragmento de video correspondiente al evento capturado en tiempo real.

La hiperventana siempre es de longitud c, sin embargo, no es un archivo estático sino que su contenido cambia de manera simultánea a la producción del archivo de video. Es decir, cuando se captura una nueva ventana de video, la última ventana contenida en la hiperventana (ventana 0) sale de este contenedor, las demás ventanas se recorren y son reenumeradas para satisfacer la condición de que la hiperventana contiene las ventanas 0 a c.

En la Figura 19 se puede observar la estructura de la hiperventana compuesta por c ventanas. Las ventanas son identificas con un subíndice i, para $0 \leq i \leq c$. A su vez dichas ventanas están conformadas por N *chunks*.

c: Representa el índice de la ventana de video que se produce de forma simultánea a la captura del evento en tiempo real.

0: Representa el índice de la ventana con el máximo retardo considerado respecto a la ventana c, es decir, es la última ventana que aún se considera como visualización en tiempo real.

Como se mencionó anteriormente, los *peers* conectados al sistema de transmisión de video en vivo, son agrupado en poblaciones y dichas poblaciones se clasifican de acuerdo con la ventana que se encuentran descargando. En el modelo desarrollado para este proyecto se representa el comportamiento (variación en el tamaño) de cada una de las poblaciones en las ventanas del video en vez de representar el comportamiento individual de cada uno de los *peers*, con la finalidad de simplificar el modelo. La clasificación mencionada es la siguiente:

X_i: Representa la población de *peers* que se encuentra descargando y reproduciendo la ventana i. Para $i \in [1, 2, 3, \dots, c-1, c]$.

X_0: Representa la población de *peers* en la ventana 0, únicamente pueden descargar el archivo de video para generar *buffer* y puedan visualizar la transmisión de video sin problemas de estancamiento en la descarga.

Con base en el análisis y revisión de los modelos para el consumo de servicio de *VoD* reportados en la literatura y al análisis de la cadena unidimensional (Erlang-B) se pudo obtener el comportamiento que tienen los usuarios en la visualización de un video y así identificar los sucesos que pueden ocurrir en la transmisión del video en vivo para comprender como estos influyen en la redimensión de las poblaciones de cada una de las N ventas que componen al archivo de video.

Los principales sucesos que se identificaron a lo largo de la descarga y reproducción de un video son:

- Conexión de un *peer*: Sucede cuando un *peer* se conecta al iniciar la descarga/reproducción del video. El *peer* deberá recibir, con mayor prioridad, recursos de un servidor *CDN* para poder descargar el video para posteriormente él compartir recursos con *peers* que inicien la descarga del video tiempo después. El *peer* no recibirá necesariamente el 100% de los recursos que necesita directamente de un servidor *CDN*, sino que puede recibir algún porcentaje de este y otro tanto de algunos otros *peers*.

- Arribo de un *peer*: Sucede cuando un *peer* se conecta al sistema para comenzar la descarga/reproducción del video. En este caso el *peer* recibirá recursos de acuerdo con el esquema de asignación de recursos, con la finalidad de recibir porciones de recursos que necesita desde diferentes *peers* habilitados para compartir recursos.

- Transferencia de un *peer* a la ventana superior: Ocurre cuando un *peer* que estaba descargando la ventana i comienza a descargar la ventana $i+1$. Por lo tanto, ya no forma parte de la población de la ventana i ahora es parte de la población que descarga la ventana $i+1$.

- Desconexión de un *peer*: Por diversas razones un *peer* puede interrumpir la visualización de un evento (mala conexión a internet, pérdida de interés en el contenido, fallas en su dispositivo, etc.) antes de que termine la transmisión. Por lo tanto, si un usuario se encontraba descargando la ventana i, se desconecta, se decrementará en 1 la población de esa ventana.

De forma general, cualquier población de una ventana contenida en $[0, N]$ se altera por la transferencia o desconexión de un *peer*. Sin embargo, el arribo de un *peer* únicamente ocurre en la ventana 0, que representa a la primera ventana del archivo de video.

La desconexión de un *peer* contempla los casos reportados en la Tabla 4 con el fin de comprender la estructura de la distribución del video.

Caso	Descripción
Primero	El *peer* que se desconectó estaba conectado directamente al servidor *CDN* y no era punto de acceso para un *peer* que llego después de iniciada la transmisión.
Segundo	El *peer* que abandona la conexión estaba conectado al servidor *CDN* y a su vez tenia a otro *peer* conectado a él.
Tercer	El *peer* que abandona la transmisión estaba conectado a otro *peer* y no tenía ningún *peer* conectado a él.
Cuarto	El *peer* que abandona el sistema estaba conectado a otro *peer* y a su vez tenía otro *peer* conectado a él.

Tabla 4. Casos de desconexión de un peer

Tomando en cuenta lo descrito anteriormente se realizó el diseño de una cadena de Markov para servicios de video en vivo y se estableció la transición que presenta la cadena en su estado, al ocurrir un suceso por parte de los usuarios conectados a la transmisión del video en vivo.

El sistema de transmisión de contenido en tiempo real se representa con la siguiente cadena de Markov:

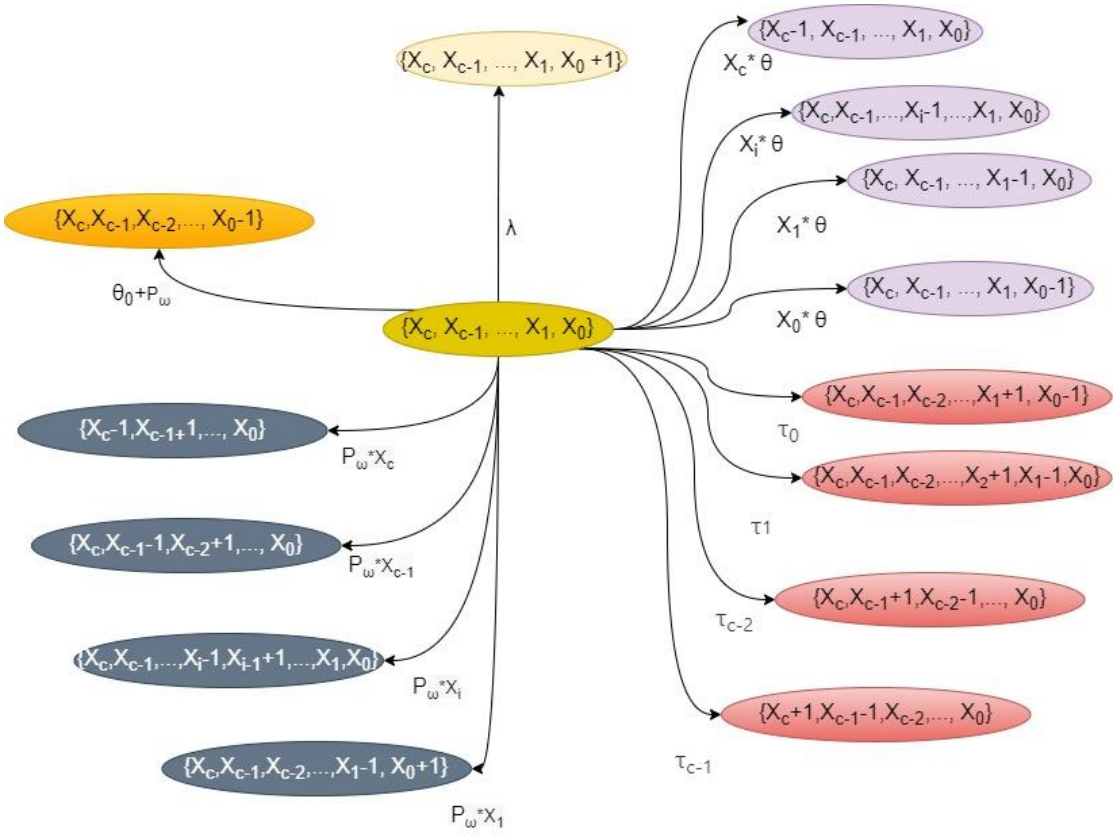


Figura 20. Cadena de Markov de un sistema de transmisión de video en vivo

En la Figura 20, se observan los sucesos que generan una transición en el estado de la cadena y el resultado de esta.

En el centro de la Figura 20, se observa al vector $\{X_c, X_{c-1}, \dots, X_1, X_0\}$, que representa al estado de la cadena de Markov para servicios de video en vivo. Este vector contiene las poblaciones de *peers* en cada una de las ventanas contenidas en la hiperventana de un archivo de video (desde la ventana 0 hasta la ventana c).

Este vector representa un estado general de la cadena de Markov, es decir, la cadena se encuentra en este estado en cualquier instante de la transmisión de video en vivo. Cada una de las poblaciones X_i, contenidas en este vector estado, tiene un valor aleatorio.

En el sistema a modelar en este proyecto se considera que los sucesos son discretos, es decir, ocurre un único evento a la vez (conexión de un usuario, transferencia de un *peer* a la ventana superior inmediata, transferencia de un *peer* a la ventana inferior inmediata o desconexión de un *peer* antes de terminar la transmisión).

A continuación, se describen los sucesos que producen un cambio en alguna(s) población de *peers* perteneciente a una ventana del archivo de video y por ende el estado de la cadena de Markov transita.

Conexión de un nuevo usuario al sistema: Una vez que inicia la transmisión de video en vivo, diversos usuarios se pueden conectar a la transmisión para visualizar la distribución de contenido en tiempo real. Dicha conexión se realiza a tasa λ, que representa la tasa de conexión de un usuario en general del sistema.

En este proyecto se supone que el usuario se debe conectar a la ventana 0 de la hiperventana para comenzar a descargar el video, crear *buffer* y así evitar congelamientos en la descarga del contenido.

Al conectarse un usuario al sistema, el estado general de la cadena de Markov transita del estado $\{X_c, X_{c-1}, \dots, X_1, X_0\}$ al estado $\{X_c, X_{c-1}, \dots, X_1, X_0 + 1\}$. Este suceso se plasma gráficamente en la parte superior de la Figura 20.

Transferencia de un *peer* a la ventana inmediata superior: Cualquier población de *peers* correspondiente a una ventana contenida en $[0, c-1]$ se modifica al generarse la transferencia de un *peer* a la ventana superior inmediata, esta transferencia ocurre a tasa τ_i, es decir, una vez que un usuario termina de descargar la ventana i y comienza a descargar la ventana $i+1$, abandona la población X_i a tasa τ_i y se agrega a la población en la ventana $i+1$ (X_{i+1}).

Debido a la naturaleza de los videos en vivo y la definición de hiperventana que se introdujo para el desarrollo de este proyecto, un *peer* que está descargando

la ventana c, al finalizar la descarga de esta, no puede ser transferido a una ventana superior inmediata; porque no hay disponible otra ventana para descargar. Es decir, la población de *peers* que se encuentra descargando la ventana actual deben esperan a que se produzca otra ventana del video y esta entre a la hiperventana para poder descargarla.

Una vez que la nueva ventana es producida e ingresa a la hiperventana, de manera automática la población X_c ahora es la población X_{c-1}. Y por lo tanto, pueden ser trasferidos a la ventana superior inmediata y comenzar la descarga de la ventana actual nuevamente.

El suceso de transferencia a la ventana superior inmediata, provoca una transición en el estado de la cadena de Markov como se describe a continuación:

$$\{X_c, X_{c-1}, \dots, X_{i+1}, X_i, \dots, X_0\} \rightarrow \{X_c, X_{c-1}, \dots, X_{i+1} + 1, X_i - 1, \dots, X_0\}$$

El suceso descrito anteriormente está plasmado gráficamente en la parte inferior de la Figura 20.

τ_i : Representa la tasa promedio de transferencia de la ventana i. Y se define con la siguiente expresión:

$$\tau_i = min\{C_\omega X_i, r_i\}\ para\ i\ \epsilon\ [0: c-1]$$

$Donde$:

C_ω: Es la tasa de descarga general de un usuario dentro del sistema.

r_i: Representa a los recursos de descarga efectivos en penuria para la ventana i, es decir, cuando la tasa de descarga es mayor que la tasa de subida el sistema entra en penuria y debe obtener recursos de la red *CDN*. Esta expresión depende del esquema de asignación de recursos, por ejemplo, el esquema de distribución uniforme que se retomará con más detalle en la siguiente sección.

Transferencia de un *peer* a la ventana inmediata inferior: Cualquier población de *peers* correspondiente a una ventana contenida en $[1, c]$ se modifica al generarse la transferencia de un *peer* a la ventana inferior inmediata a tasa $P_\omega X_i$, es decir, cuando un usuario por diversas razones (fallas en sus servicios, fallas en sus dispositivos, etc.) deja de descargar el video en vivo a la misma tasa que se está capturando el evento en tiempo real y produciendo una nueva ventana del video, por lo tanto el *peer* se atrasa en la descarga del video y pasa de la ventana i (abandona la población X_i) a la ventana $i-1$ (se adiere a la población X_{i-1}).

Por la naturaleza de los videos en vivo y la definición de hiperventana, un *peer* que está descargando la ventana 0 si interrumpe su proceso de descarga, no puede ser transferido a una ventana inferior inmediata; porque esto provoca que salga de la hiperventana y por lo tanto abandone el sistema. Este caso en particular representa un factor para generar la desconexión de un usuario en la

ventana, por ello, su efecto en el estado de la cadena de Markov se hace en conjunto a la desconexión de un usuario en la ventana 0, que se describirá más adelante.

El suceso de transferencia a la ventana inferior inmediata provoca una transición en el estado de la cadena de Markov como se describe a continuación:

$$\{X_c, X_{c-1}, \dots, X_i, X_{i-1}, \dots, X_0\} \rightarrow \{X_c, X_{c-1}, \dots, X_i - 1, X_{i-1} + 1, \dots, X_0\}$$

El suceso descrito anteriormente está plasmado gráficamente en la parte izquierda de la Figura 20.

Desconexión de un *peer* antes de terminar la transmisión: Por causas diversas (fallas de conexión, fallas en sus servicios, fallas en sus dispositivos, desinterés en el contenido, etc.) un *peer* que está visualizando la transmisión de video en vivo puede desconectarse.

Entonces, cualquier población de *peers* de la cadena cambia cuando se genera la desconexión de un *peer* a tasa $X_i\theta$, antes de que finalice la transmisión en vivo. $X_i\theta$ representa la tasa promedio de desconexión de la población que se encuentra descargando ventana i. Es el resultado de multiplicar la población de la ventana i (X_i) por θ (tasa de desconexión de un *peer* conectado al sistema en general).

Este suceso provoca una transición en el estado de la cadena de Markov como se describe a continuación:

$$\{X_c, X_{c-1}, \dots, X_i, \dots, X_0\} \rightarrow \{X_c, X_{c-1}, \dots, X_i - 1, \dots, X_0\}$$

Lo anterior quiere decir que un *peer* que formaba parte de la población X_i por alguna razón abandono el sistema antes de finalizar la transmisión en vivo, por lo tanto, la población de la ventana i se decrementa en 1. Este evento es representado en la sección derecha de la Figura 20.

Los *peers* conectados a la ventana 0 son más susceptibles a desconectarse del sistema, en caso de interrumpir su proceso de descarga, se generó una nueva ventana del video, por una mala conexión a internet, problemas con el hardware, no tener óptimas *QoE ó* QoS. Por lo tanto, se considera a θ_0 como la tasa de desconexión de los *peers* pertenecientes a la población X_0 y se define como:

$$\theta_0 = \theta + P_\omega$$

Donde:

θ: Representa la tasa de abandono general de un usuario dentro del sistema

P_ω: Representa la tasa a la cuál es producido el archivo de video.

Una vez diseñada la cadena de Markov y definidos los sucesos que generan un cambio en su estado, se diagramo la solución matemática a implementar para simular la ocurrencia de los sucesos y la transición de estado en la cadena.

Una vez diseñada la cadena de Markov y definidos los sucesos que generan un cambio en su estado, se diagramo la solución matemática a implementar para simular la ocurrencia de los sucesos y la transición de estado en la cadena.

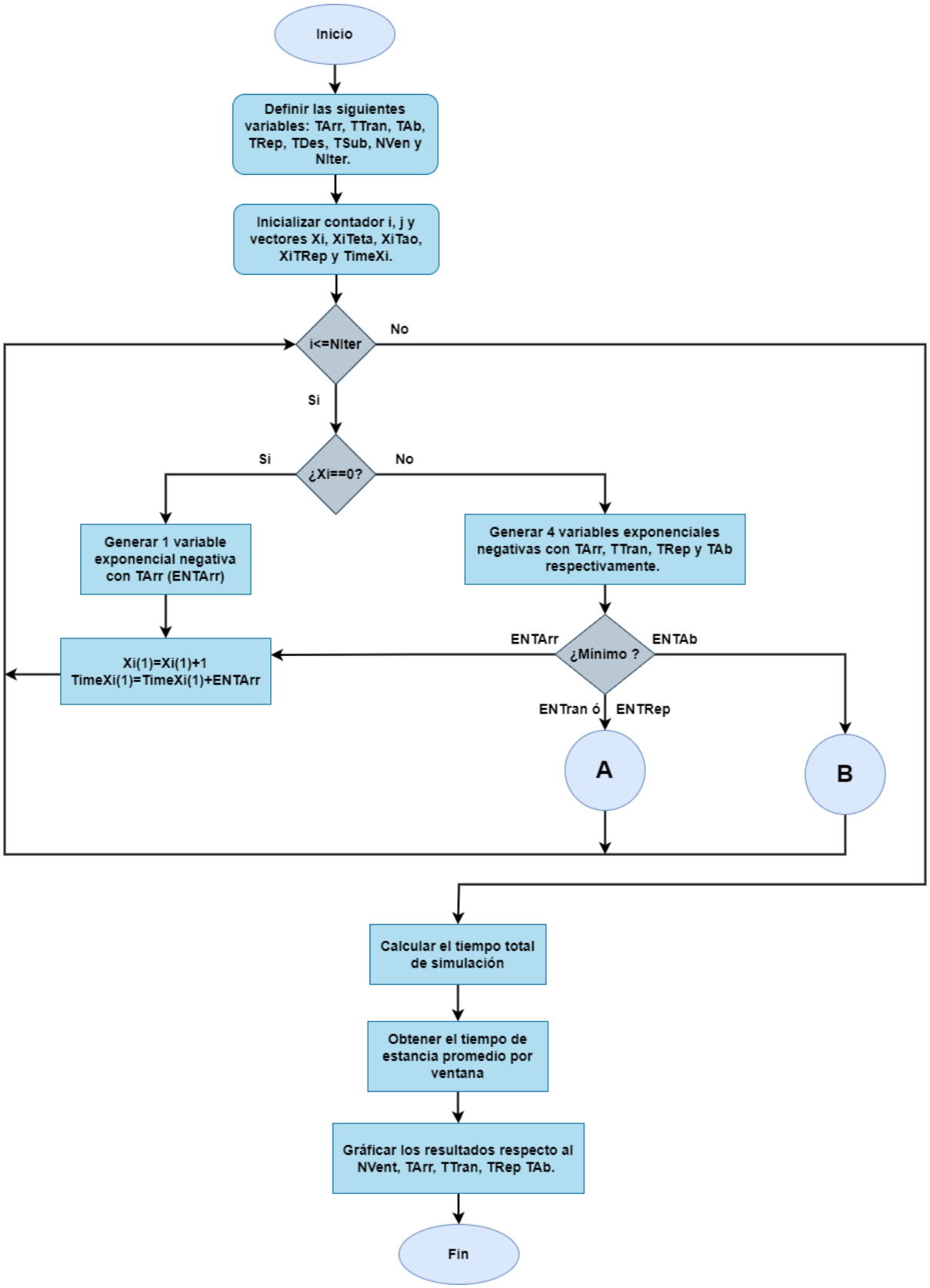


Figura 21. Diagrama de flujo de solución matemática para la cadena de Markov de un sistema de video en vivo

En el diagrama de la Figura 21 se encuentra el algoritmo que se debe implementar para dar solución por implementación a la cadena de Markov que representa un sistema de servicio de video en vivo. En primer lugar, se deben definir los parámetros de entrada para simular el comportamiento de los usuarios dentro del sistema y el cambio que resulta en el estado de la cadena.

Las tasas que se deben definir son la tasa de arribo, tasa de abandono, tasa de descarga y tasa de subida. La tasa de transferencia depende del esquema de asignación de recursos y se definirá con más detalle en las secciones próximas.

Otro valor importante para la implementación son el número de ventanas, que representa la longitud de la hiperventana. Y finalmente se debe definir el número de iteraciones que se desea ejecutar la simulación.

Después de eso se inicializan los contadores i y j, que nos servirán para avanzar en las iteraciones y recorrer vectores respectivamente. En este paso igual se inicializan los vectores *Xi* (poblaciones por ventana de la hiperventana), *Xiteta* (escalamiento de las poblaciones por la tasa de abandono), *Xitao* (escalamiento de las poblaciones por la tasa de transferencia), *XiTRep* (escalamiento de las poblaciones por la tasa de reproducción) y *TimeXi* (Tiempo promedio de estancia por cada una de las ventanas de la hiperventana).

Posteriormente, se inician las iteraciones. Se comienza por verificar si las poblaciones en las ventanas son cero, en caso de que sea cierto, el único evento que se puede generar es la conexión de un usuario al sistema. Por lo tanto, se genera una variable aleatoria con distribución exponencial negativa y la tasa de arribos, se incrementa en una unidad la población de la ventana 0 y se almacena *TimeXi* el valor de la variable aleatoria y se retorna a la condición de iteración.

En caso de que las poblaciones del vector *Xi* sean diferentes de cero, se deben generar 4 variables aleatorias con distribución exponencial negativa y tasas de arribo, de abandono, de reproducción y de transferencia. Esto con la finalidad de conocer que suceso ocurrió (conexión, desconexión, transferencia a la ventana inferior o transferencia a la ventana superior). La variable con el valor mínimo es la que indicara que suceso ocurrió.

Si la de menor valor fue la generada a partir de la tasa de arribo se incrementa en una unidad la población de la ventana 0 y se almacena *TimeXi* el valor de la variable aleatoria y se retorna a la condición de iteración. Y en un caso diferente se prosigue a los procesos A, cuando la de menor valor es la variable generada con la tasa de transferencia o la tasa de reproducción. O bien se sigue el proceso B, cuando la de menor valor es generada con la tasa de abandono.

Finalmente, cuando las iteraciones terminan se obtienen los arribos totales por ventana (poblaciones de Xi), el tiempo promedio de estancia por ventana y se grafican los resultados en función a los parámetros de entrada.

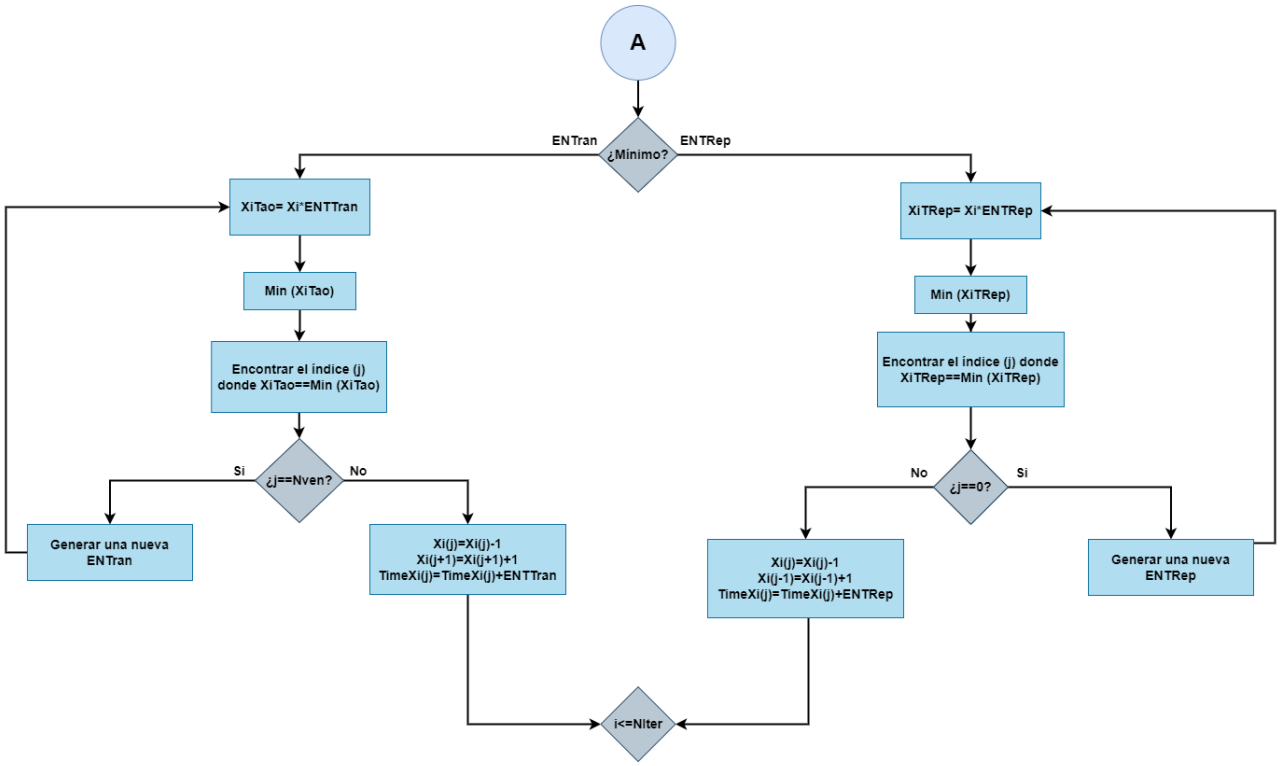


Figura 22. Diagrama para mín=ENTran ó EnTRep

En el diagrama de la Figura 22 se representa el algoritmo de proceso A, cuando la variable aleatoria de menor valor es la variable generada con la tasa de transferencia o la tasa de reproducción.

Se realiza la comparación para saber cuál fue la de valor mínimo.

- Variable aleatoria generada a partir de la tasa de transferencia: se escalan las poblaciones de cada una de las ventanas pertenecientes a la hiperventana por la variable aleatoria generada. Posteriormente, se obtiene el mínimo de ese vector escalado con la finalidad de conocer el índice de la ventana donde ocurrió la transferencia a la ventana superior.

  Una vez que se conoce el índice, se verifica si es igual a la longitud de la hiperventana. En caso de ser verdadero, se genera una nueva variable aleatoria y se escala nuevamente el vector de poblaciones por esta nueva variable. Debido a que un usuario en la ventana c, no puede avanzar hacia otra ventana puesto que se encuentra en sincronía con la captura del evento en vivo.

  En caso de ser falso se decrementa en una unidad la población de la ventana j, se incrementa en una unidad la población de la ventana $j + 1$ y se almacena en *TimeXi* el valor de la variable aleatoria. Finalmente se retorna a la condición de iteración.

- Variable aleatoria generada a partir de la tasa de producción: se escalan las poblaciones de cada una de las ventanas pertenecientes a la hiperventana por la variable aleatoria generada. Posteriormente, se obtiene el mínimo de ese vector escalado con la finalidad de conocer el índice de la ventana donde ocurrió la transferencia a la ventana inferior.

  Una vez que se conoce el índice, se verifica si es igual a 0. En caso de ser verdadero, se genera una nueva variable aleatoria y se escala nuevamente el vector de poblaciones por esta nueva variable. Debido a que un usuario en la ventana 0, no puede retroceder hacia una ventana inferior puesto que se encuentra en el de la hiperventana (visualización en tiempo real), el resultado es decrementar esa población y tomarlo como un abandono de la ventana 0.

  En caso de ser falso se decrementa en una unidad la población de la ventana j, se incrementa en una unidad la población de la ventana $j - 1$ y se almacena en *TimeXi* el valor de la variable aleatoria. Finalmente se retorna a la condición de iteración.

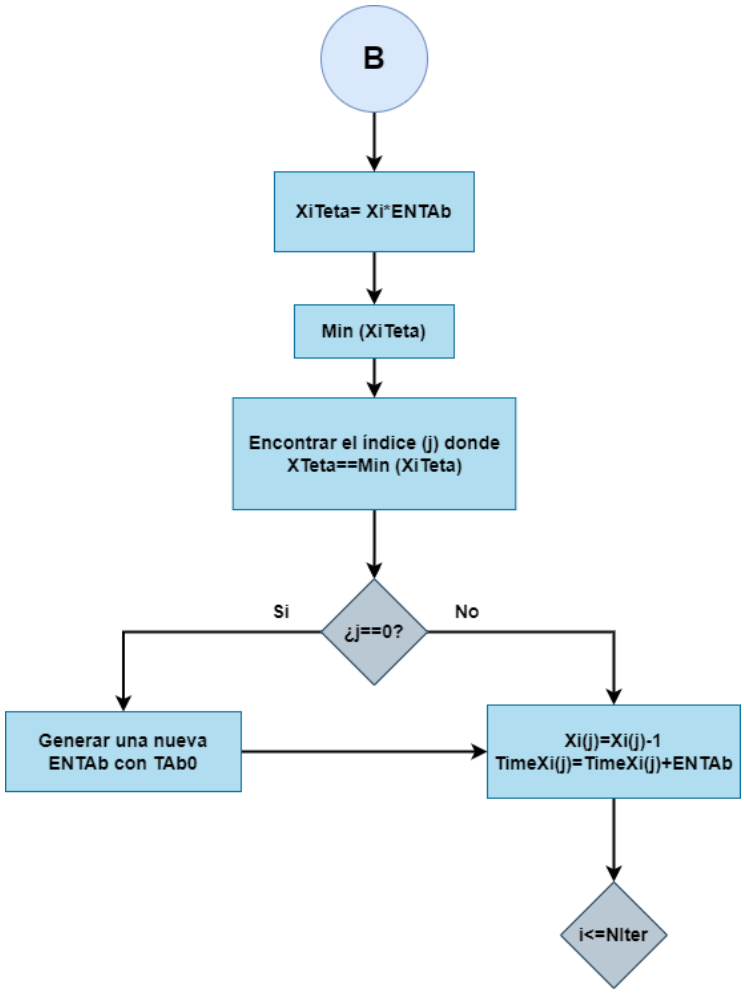


Figura 23. Diagrama para mín=ENTAb

En el diagrama de la Figura 23 se representa el algoritmo de proceso B, cuando la variable aleatoria de menor valor es la variable generada con la tasa de abandono.

Se escalan las poblaciones de cada una de las ventanas pertenecientes a la hiperventana por la variable aleatoria generada. Posteriormente, se obtiene el mínimo de ese vector escalado con la finalidad de conocer el índice de la ventana donde ocurrió la desconexión.

Una vez que se conoce el índice, se verifica si es igual a 0. En caso de ser verdadero, se genera una nueva variable aleatoria a partir de la tasa de abandono de la ventana 0. Debido a que en el presente trabajo se suponen que un usuario en la ventana 0, abandona el sistema a una tasa de abandono promedio diferente a la tasa de abandono en las demás ventanas.

Posteriormente, se decrementa en una unidad la población de la ventana 0 y se almacena en *TimeXi* el valor de la variable aleatoria.

En caso de que el índice sea diferente de cero se decrementa en una unidad la población de la ventana j, y se almacena en *TimeXi* el valor de la variable aleatoria. Finalmente se retorna a la condición de iteración.

En el diagrama de la Figura 21 se presenta a la variable $\tau(TTran)$, cuyos valores se definen a partir de un esquema de asignación de recursos que se retoma de forma detallada en la sección próxima. Las estadísticas que se obtendrán de la implementación del diagrama se graficarán en función de distintos valores para los parámetros de entrada $(c, P_\omega, \mu_\omega, \theta, etc.)$

5.2 Diseño del esquema de asignación de recursos para servicios de video en vivo

En esta sección se desarrolla el bosquejo del esquema de asignación de recursos para servicios de video en vivo. En este diseño se explican las condiciones que debe cumplir el sistema para realizar la asignación de recursos. En esta etapa se ha desarrollado un esquema de asignación uniforme y considerando condición de abundancia dentro del sistema.

Tomando como base el análisis del comportamiento de los *peers* se plantea desarrollar un esquema de asignación de recursos conveniente para los servicios de video en vivo. Los esquemas presentados en [7] (Q ventanas hacia atrás) y [12] (GDPV) y muestran un desempeño eficiente para la asignación de recursos para servicios de video bajo demanda (*VoD*). Sin embargo, tomando en

cuenta los objetivos de este proyecto con un enfoque dirigido a los servicios de video en vivo, se plantea realizar una adecuación en primera instancia al esquema de asignación de recursos de distribución uniforme, sin dejar de lado la posibilidad de retomar Q ventanas hacia atrás o GDPV para ser aplicados a este tipo de servicios.

En primer lugar, se analizó la cantidad de recursos necesarios para que la población correspondiente a una ventana sea atendida y pueda descargar de manera satisfactoria el video en vivo. Con el propósito de establecer una expresión que permita cuantificar esta cantidad de recursos al cual se le denomino ancho de banda consumido.

De manera general dentro del sistema se tiene la tasa de descarga global que regula el ancho de banda consumido en el proceso de descarga del contenido en vivo, dicha tasa es representada con c_ω. Esta tasa permite conocer la máxima velocidad a la que un *peer* en general conectado al sistema realiza la descarga del contenido. Sin embargo, esta tasa no específica la cantidad de recursos necesarios para que la población de una ventana descargue de forma continua el contenido de la ventana posterior.

Por lo tanto, se establece a $c_\omega X_i$ como la cantidad de recursos (ancho de banda de descarga) requerida por la población en la ventana i para descargar el video en vivo. La cantidad $c_\omega X_i$ es el resultado de multiplicar la tasa global c_ω, que es la tasa individual máxima de descarga, por X_i (la población de *peers* en la ventana i). De igual forma dentro del sistema existe la tasa de producción, la cual indica la velocidad a la cual se captura el evento en vivo y se genera el archivo de video, es decir, esta tasa indica la velocidad con la que los usuarios deben descargar el video para no perder sincronía respecto a la ventana actual (c). Esta tasa es denotada por P_ω.

Para el presente proyecto se supone que la tasa de producción debe ser menor que la tasa de descarga $(C_\omega > P_\omega)$ para evitar que un usuario pierda sincronía con la producción del video, sea transferido a la ventana inferior inmediata y/o eliminado del sistema.

Una vez que se analizó la cantidad de recursos necesarios para que la población correspondiente a una ventana en específico pudiera descargar el video en vivo se analizó el tamaño de la población de las ventanas superiores para conocer si la cantidad de recursos requeridos puede ser cubierta únicamente por los recursos procedentes de los *peers* o se deben solicitar recursos a los servidores de la red *CDN*.

En el sistema de servicios de video en vivo se busca que los recursos provenientes de la red *P2P (peers)* sean asignados en primer lugar y en caso de requerir más recursos, asignar la cantidad faltante de la red *CDN* (servidores).

Dentro del sistema, para poder realizar la asignación de recursos se deben considerar dos casos principales:

El primero de ellos es cuando la cantidad de recursos requeridos por una población $(c_\omega X_i)$ para ser atendidos es menor a la cantidad de recursos disponibles provenientes de los *peers* y de los servidores *CDN*. Por lo tanto, el existe condición de abundancia y los *downloaders* descargan el video a la velocidad requerida.

Bajo condición de abundancia, se necesita calcular la cantidad de recursos que proporciono la red *P2P* y la cantidad de recursos que proporciono la red *CDN*, con la finalidad de conocer si el sistema fue autónomo, es decir, únicamente basto con los recursos provenientes de los *peers* para atender a los *downloaders*. O bien, si fue necesario solicitar recursos de la red *CDN.*

El segundo caso, es cuando la cantidad de recursos requeridos por una población $(c_\omega X_i)$ es mayor a la cantidad de recursos ofrecidos por la red *P2P* y la red *CDN*, en este caso, dentro del sistema existe condición de penuria. Y por lo tanto, los recursos provenientes tanto de la red *CDN* como de la red *P2P* son insuficientes para que los *downloaers* descarguen el video a la velocidad requerida.

Si el sistema, se encuentra en penuria, *los downloaders* pierden sincronía con la producción del video, se atrasan y finalmente son eliminados del sistema, y como resultado el sistema colapsa.

A razón del segundo caso, es que se decide trabajar con condición de abundancia para modelar de manera más cercana a la realidad los eventos realizados por los usuarios finales al momento de descargar y visualizar un archivo de video.

Dentro del sistema que se desarrolla en el proyecto, los *peers* conectados a la transmisión en directo reciben también el nombre de *downloaders*. Debido a la naturaleza de la red *P2P* los *downloaders*, son parte medular del ancho de banda de subida que requiere el sistema, ya que los *peers* tienen la capacidad de compartir el contenido almacenado en su *buffer* a otros *peers* que se encuentren descargando el video en ventanas inferiores dentro del sistema. Este hecho permite que el ancho de banda requerido por una población no sea consumido en su totalidad de la red *CDN* y así se evita que la red se sature y el sistema colapse, por lo tanto, los servidores atienden de manera equitativa a los *downloaers* situados en las diferentes ventanas que componen a la hiperventana, de acuerdo con la distribución uniforme.

Sin embargo, el sistema puede distribuir los recursos provenientes de la red *CDN* de manera tal, que algunos *peers* reciban mayor cantidad de recursos de estos servidores. Y así evitar que algunas poblaciones tengan mayor disponibilidad de recursos para ser atendidas que otras. Como se ha mencionado anteriormente, en primer lugar se asignan los recursos provenientes de la red P2P y en caso de

ser necesario se debe proporcionar un espacio del ancho de banda proveniente de la red *CDN*.

El sistema tiene una tasa de subida global que establece el ancho de banda de subida promedio que puede proporcionar un *peer* en el proceso de la transmisión de contenido en vivo, dicha tasa es representada con μ_ω. Esta tasa permite conocer la velocidad promedio a la que un *peer* conectado al sistema sube la porción del archivo de video que tiene almacenada en su *buffer*.

Sin embargo, μ_ω no específica la cantidad de recursos que puede proporcionar la población de la ventana i a los *peers* en ventanas $j,\ 0 \leq\ j < i - 1$, para que puedan comenzar el proceso de descarga del archivo de video.

Por lo tanto, se establece a $\mu_\omega X_i$ como la cantidad de recursos (ancho de banda de subida) proporcionada por la población en la ventana i. Es decir, los *chunks* contenidos en su *buffer* para compartir con otros *peers* en ventanas inferiores. La cantidad $\mu_\omega X_i$ es el resultado de multiplicar la tasa global μ_ω, que es la tasa individual promedio de subida, por X_i la población de *peers* en la ventana i.

El objetivo principal de un esquema de asignación de recursos es procurar condiciones de abundancia dentro del sistema, es decir, que los recursos disponibles en la red (ancho de banda de subida) sea mayor o igual al demandado por el sistema (ancho de banda de descarga) y que este a su vez sea proporcionado equitativamente a todos los *peers* conectados al sistema.

Con base en lo descrito hasta este punto de la investigación se decide desarrollar un bosquejo del esquema de asignación de recursos para servicios de video en vivo basado en el esquema de distribución uniforme.

En la Figura 24, se observa que la población de la ventana i puede ser atendida (obtener recursos) de poblaciones en ventanas superiores, es decir, la población en la ventana i puede ser atendida por poblaciones en ventanas $k, i + 1 \leq k \leq c$.

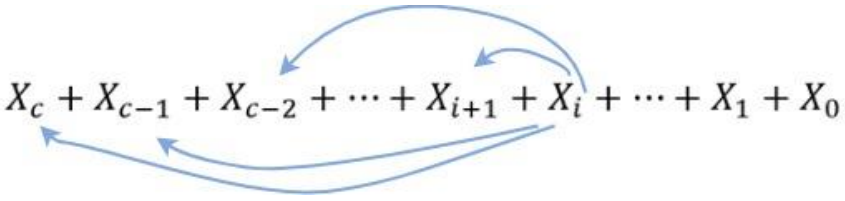


Figura 24. Obtención de recursos para la población en la ventana i

En la Figura 24 se observa que la población de una ventana cual esta sea puede obtener recursos únicamente de poblaciones situadas en ventanas superiores. Esto debido a que los usuarios en ventanas superiores tienen un mayor progreso en la descarga del contenido en vivo.

Esquema de Asignación de Recursos Uniforme para servicios de video en vivo

Inicio

1.- Arriba o es transferido un usuario a la ventana $i, i = 0, 1, 2, \dots, c-1$ de la hiperventana
2.- Se define $k = i + 1$
3.- Repetición de Recolección
Se almacenan las poblaciones de *peers* situadas en las ventanas $k, i+1 \leq k \leq c$, que pueden atender a los usuarios en la ventana i.
Fin de Repetición
4.- Conjunto de poblaciones de *peers* con recursos disponibles para atender a la población en la ventana i.

5.- ¿La tasa real de descarga es igual a la tasa máxima de descarga?

Si: Asignar los recursos

No: Se asignan recursos, ¿aún hay *peers* en la lista?
Si: Retornar a paso 4
No: Buscar recursos en el *CDN* y repetir condición Si del paso 5
6.- Generar estadísticas

Fin

Debido a que el esquema de asignación de recursos en esta etapa del proyecto es un esquema uniforme, el número de ventanas de donde provienen los recursos no está limitado, es decir, la población en la ventana i pueden obtener recursos de ventanas k tal que $k,\ i+1 \leq k \leq c$, además de obtener recursos provenientes de los servidores *CDN,* como se muestra en la siguiente expresión:

$$R_i = \left((\mu_\omega * X_{i+1})\left(\frac{X_i}{X_0 + X_1 + \cdots + X_i}\right) + \mu_s\left(\frac{X_i}{X_0 + X_1 + \cdots + X_c}\right)\right) +$$

$$\left((\mu_\omega * X_{i+2})\left(\frac{X_i}{X_0 + X_1 + \cdots + X_{i+1}}\right) + \mu_s\left(\frac{X_i}{X_0 + X_1 + \cdots + X_c}\right)\right) + \cdots +$$

$$\left((\mu_\omega * X_{c-1})\left(\frac{X_i}{X_0 + X_1 + \cdots + X_{c-2}}\right) + \mu_s\left(\frac{X_i}{X_0 + X_1 + \cdots + X_c}\right)\right) +$$

$$\left((\mu_\omega * X_c)\left(\frac{X_i}{X_0 + X_1 + \cdots + X_{c-1}}\right) + \mu_s\left(\frac{X_i}{X_0 + X_1 + \cdots + X_c}\right)\right)$$

Realizando análisis matemático se observa que X_i, μ_ω y μ_s son factores en común en las expresiones sumadas, por lo tanto, esa suma se puede resumir y generalizar para cualquier población como se muestra en la siguiente expresión:

$$R_i = X_i \left(\sum_{k=i+1}^{c} \frac{\mu_\omega X_k(t)}{\sum_{j=0}^{k-1} X_j(t)} + \frac{\mu_s}{\sum_{j=0}^{c} X_j(t)} \right)$$

Capítulo 6

Implementación

Para iniciar el desarrollo e implementación del código fuente, que permite realizar la evaluación por implementación de un sistema de transmisión de video en vivo, se siguió el algoritmo mostrado en la Figura 21. En primera instancia, se aplico este algoritmo para servicios de video bajo demanda *(VoD)* abordados en [5]. Esto con la finalidad de obtener resultados que fueron aprobados, validados y publicados acerca del funcionamiento de un sistema de transmisión de video y así poder tomar estos resultados como marco de referencia para conocer los resultados esperados que se deben mostrar al implementar la evaluación de un sistema *livestreaming.*

Para la primera gráfica presentada a continuación se trabajó con el esquema de asignación de recursos de distribución uniforme (DU), en la cual se observa el número de downloaders (*leeches*) en el sistema de video bajo demanda. En esta primera gráfica se observa el número promedio de *downloaders* en un archivo de video con N=24.

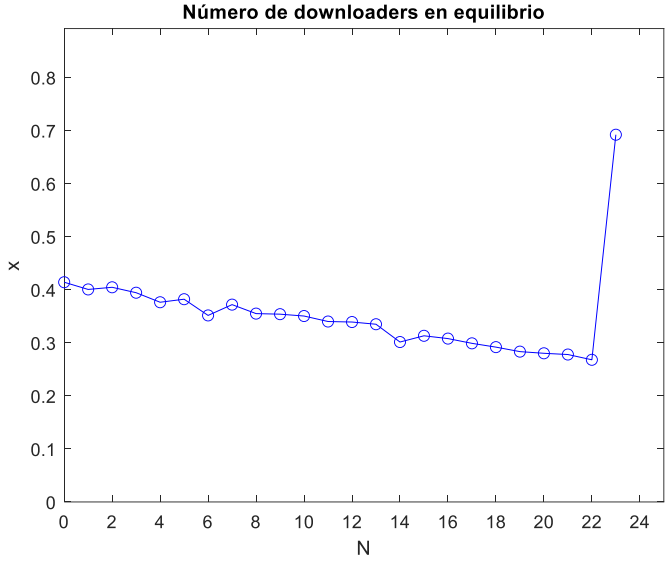


Figura 25. Gráfica bidimensional de downloaders promedio en VoD bajo DU

Una vez obtenida la gráfica de poblaciones promedio para un archivo de N ventanas, se procedió a obtener las poblaciones de *downloaders* por ventana dentro de un archivo de video variando el número de ventanas N y la tasa de desconexión del sistema.

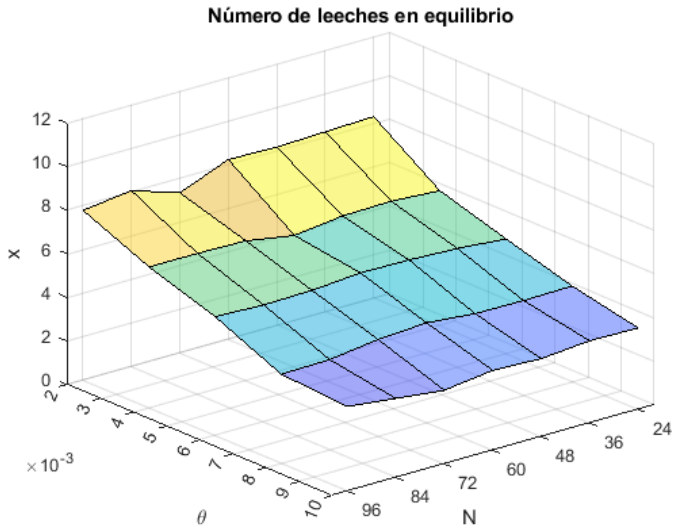


Figura 26. Gráfica de numero de downloaders en VoD bajo DU

Una vez analizada la gráfica en la Figura 26 y observar los resultados además de cambiar los valores de algunos parámetros y verificar si se comportaba de manera similar se siguió trabajando en el programa para poder obtener la gráfica del número de seeds para el sistema de video bajo demanda usando el mismo esquema de asignación de recursos (DU).

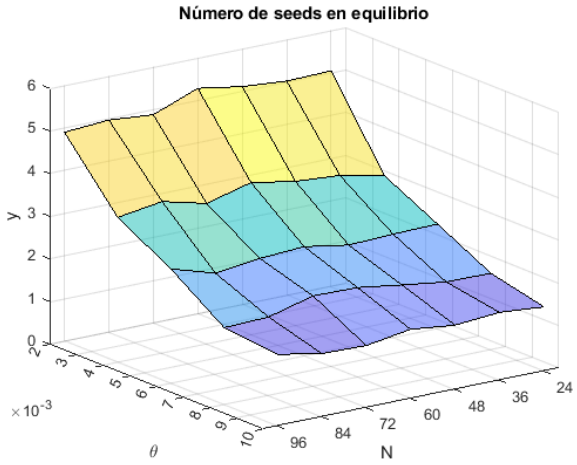


Figura 27. Gráfica de numero de seeds en VoD bajo DU

Después de obtener las gráficas arriba mencionadas y compararlas con las que se describen en los textos para video bajo demanda y observar que tenemos resultados muy similares se dice que nuestro programa está funcionando adecuadamente, ahora bien se procede a realizar cambios, adecuaciones al programa y utilizando el mismo esquema de distribución uniforme con la diferencia de que ahora ya será para los servicios de video en vivo (*live streaming*) para observar el comportamiento y poder comparar nuevamente con las de video bajo demanda VoD.

Continuando con la siguiente gráfica observamos las poblaciones promedio de downloaders que se tienen para *live streaming* con el esquema DU.

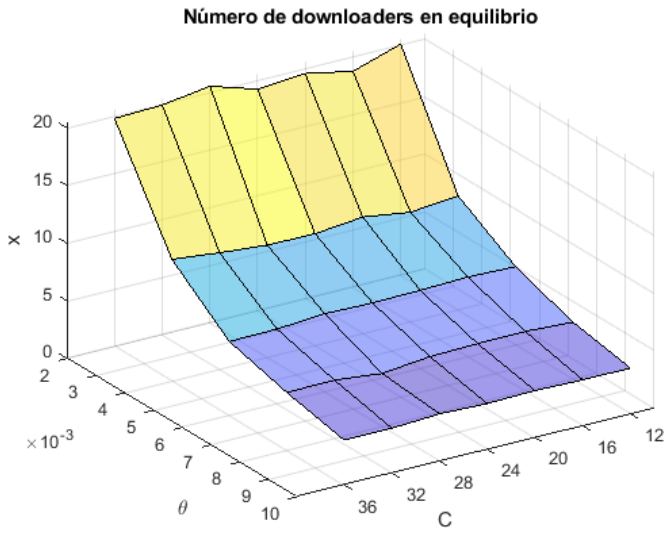


Figura 28. Gráfica poblaciones promedio de downloaders live stream DU

Se comparan las gráficas 26 y 28, si se observa especialmente el eje de las X (poblaciones de downloaders) tenemos un aumento considerable en la 26 que es la que trata el video en vivo.

Una vez que se realizó la comparación de las gráficas VoD y *live streaming* seguimos en marcha con la programación y ahora se incorpora otro estado de la cadena de Markov, transferencia a la ventana inferior (TVI), nuevamente obtenemos la gráfica de las poblaciones de downloaders para *live streaming* con el esquema DU considerando la nueva transición de la cadena.

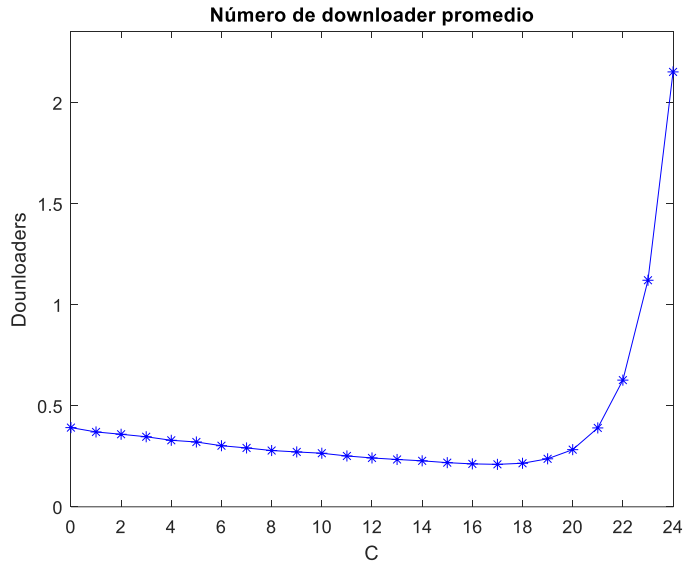


Figura 29. Gráfica bidimensional de downloaders promedio en VoD bajo DU

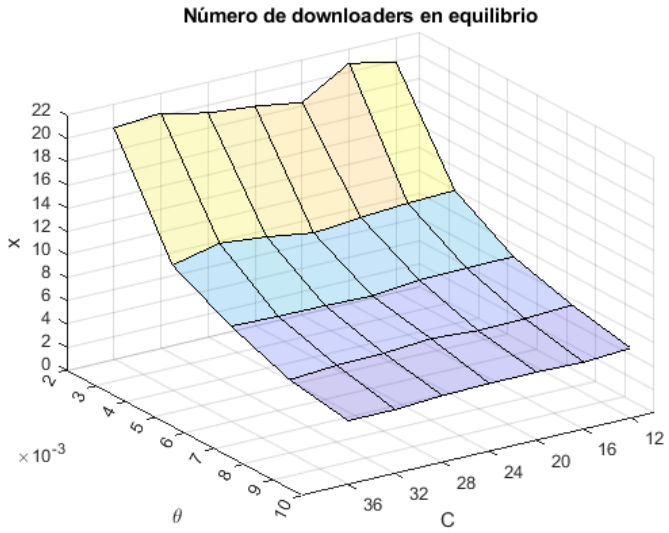


Figura 30. Gráfica downloaders promedio live stream DU TVI

Después de tener programado el estado de la cadena transferencia a la ventana inferior (TVI), se realizaron pruebas variando un parámetro diferente a la vez propio del programa. Las siguientes tres gráficas representan las poblaciones de downloaders promedio en el sistema de *live streaming* considerando ya el evento TVI.

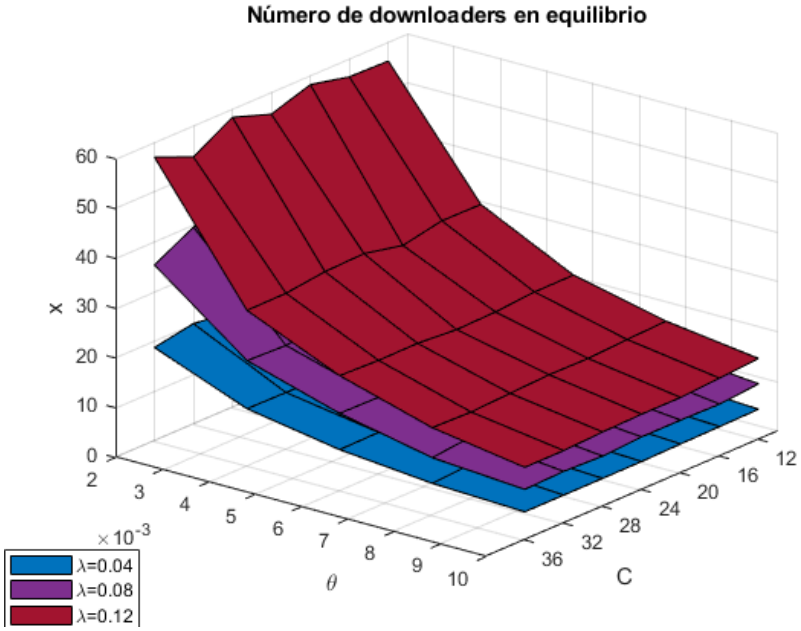


Figura 31. Gráfica de downloaders promedio variando λ

En la Figura 28 se utilizaron los valores para $\lambda = 0.04, 0.08, 0.12\ respectivamente$, se puede observar cómo es que las poblaciones de downloaders incrementan de forma reciproca al aumento de la tasa λ, es decir, cuando escalamos el valor de esta tasa de arribo las poblaciones muestran un incremento en igual magnitud.

Recordando que λ representa el número de arribos de los usuarios al sistema, se especula que al variar esta tasa exista un incremento proporcional en el tamaño de las poblaciones promedio de downloaders. En la figura 28, se aprecia que cuando el valor de λ es duplicado y triplicado, respectivamente, el tamaño de las poblaciones promedio de downloaders sufre un incremento de la misma magnitud.

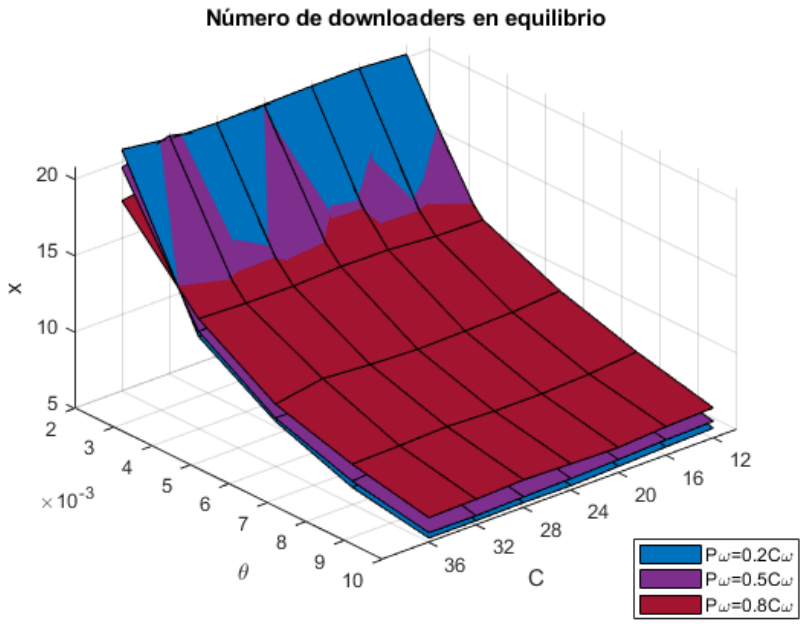


Figura 32. Gráfica de downloaders promedio variando $P\omega(tasa\ de\ reproducción)$

En la gráfica anterior se modificó el valor de la tasa de reproducción del video $P\omega = 0.2C\omega, 0.5C\omega, 0.8C\omega\ respecticamente$, es decir, se modifica la velocidad a la que se produce el video y por ende la velocidad a la que los usuarios deben visualizar el video para evitar un retraso en la visualización. En este sentido, los usuarios que no descarguen el video a tasa P_ω presentan un atraso en la reproducción y por lo tanto son transferidos a la ventana inferior inmediata.

En la Figura 29 se representa este evento y se puede observar que cuando $P_\omega \ll C_\omega$ y la tasa de abandono θ es pequeña el número de *downloaders* es alto, pero cae precipitadamente cuando θ aumenta. Por otro lado cuando P_ω es la mitad que C_ω los *downloaders* descargan el video de manera más estable y no ocurre una diferencia abrupta entre los valores de poblaciones de *downloaders* promedio.

Finalmente, cuando $P_\omega \cong C_\omega$ el abandono de usuarios va en función de la tasa de abandono θ y no muestran una gran diferencia entre las poblaciones promedio de *downloaders.*

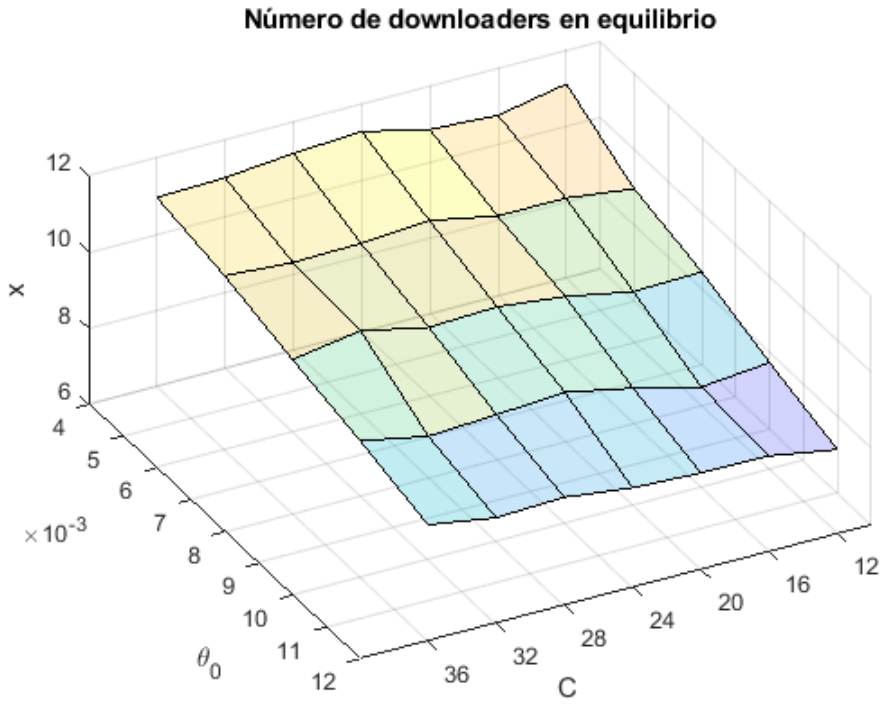


Figura 33. Gráfica de downloaders promedio variando θ

El siguiente paso es ahora observar el ancho de banda que está consumiendo el sistema dentro del sistema de *live streaming* y para ello tenemos la siguiente grafica que es el primer acercamiento que se hizo, en el entendido de que el ancho de banda se calcula multiplicando a la población promedio de cada ventana (X_i) por la tasa C_ω.

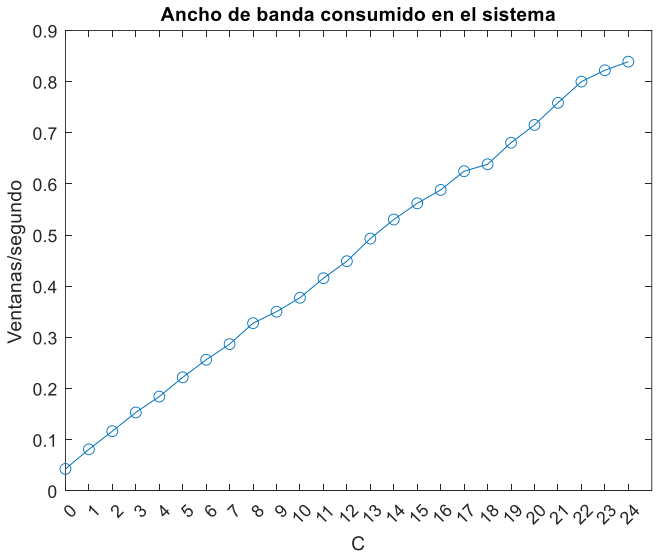


Figura 34. Ancho de banda que demanda el sistema

Las siguientes dos gráficas que vienen a continuación se obtienen a partir del tiempo promedio y representan el tiempo promedio de descarga de una ventana en el sistema de live streaming. La Figura 32 es para las ventanas $1 - C$ y la Figura 33 es para la ventana 0. Ya que se obtiene con la ecuación $T_0 = \frac{1}{C_0+\theta}$ la tasa de abandono es diferente para la ventana 0 y las demás ventanas.

Para obtener estas gráficas se incluyeron ya los recursos de los servidores (μ_s). Mediante la siguiente ecuación $\left(\frac{1}{1+2^i}\right) * \mu_s$

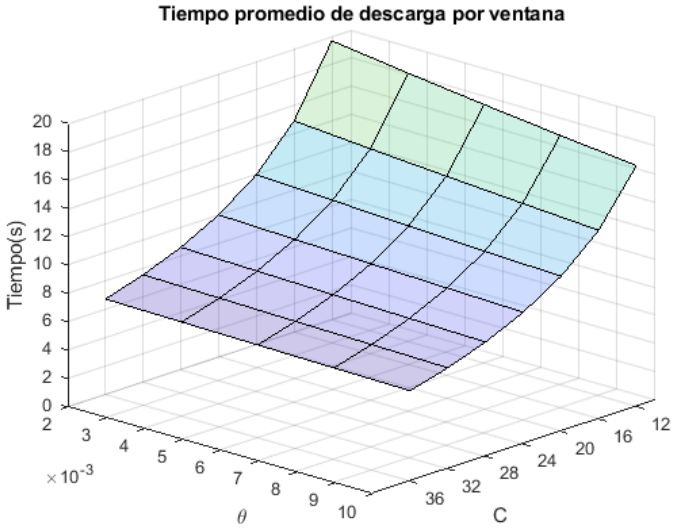


Figura 35. Tiempo promedio de descarga por ventana

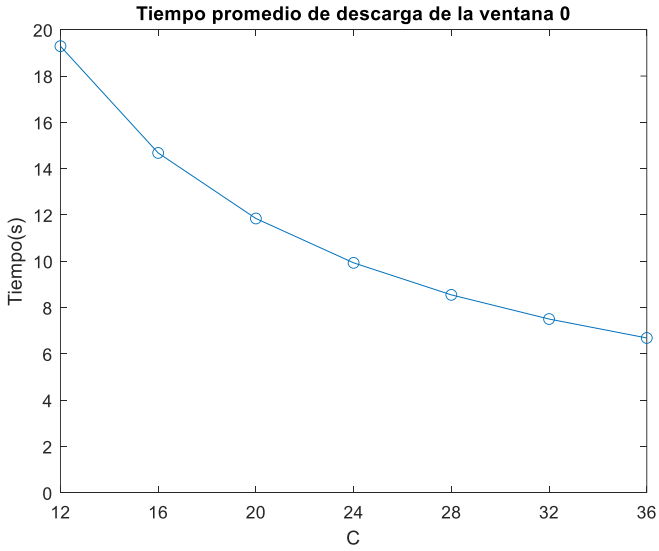


Figura 36. Tiempo promedio de descarga ventana 0

Una vez que se tiene una nueva versión de la ecuación se obtiene nuevamente la gráfica de las poblaciones promedio.

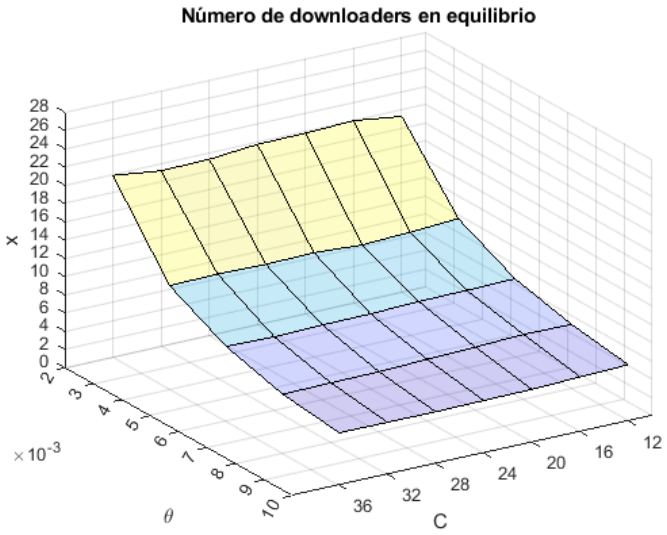


Figura 37. Poblaciones promedio de downloaders con nueva expresión

La figura anterior representa las poblaciones promedio de downloaders en el sistema *live streaming*, considerando los recursos de servidores ms. Igual se consideran con la ecuación anterior: $\left(\frac{1}{1+2^i}\right) * \mu_s$. A esto se le suma la siguiente ecuación $\tau_{i,i+1} = \min\left\{C_\omega X_i \mu_\omega \left(X_i \sum_{k=i+1}^{C} \frac{X_k}{\sum_{j=0}^{k-1} X_j} + m\frac{X_c}{X}\right)\right\}$ $donde\ X = \sum_{k=0}^{C} X_k$ tomada de [5].

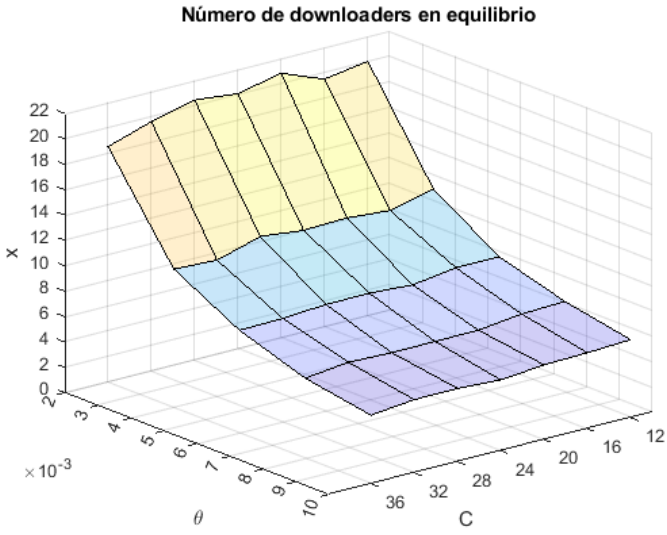


Figura 38. Downloaders promedio considerando μ_s

La figura 35 representa las poblaciones promedio de downloaders en el sistema de *live streaming*, considerando los recursos de servidores μ_s y considerando que hay transición en la última ventana, pero en vez de moverse a una ventana más hacia arriba, las poblaciones quedan igual y lo único que aumenta es el tiempo.

Anteriormente se obtuvo el ancho de banda que demanda el sistema, a continuación, tenemos el ancho de banda que consumen las poblaciones de *peers* directamente de los servidores.

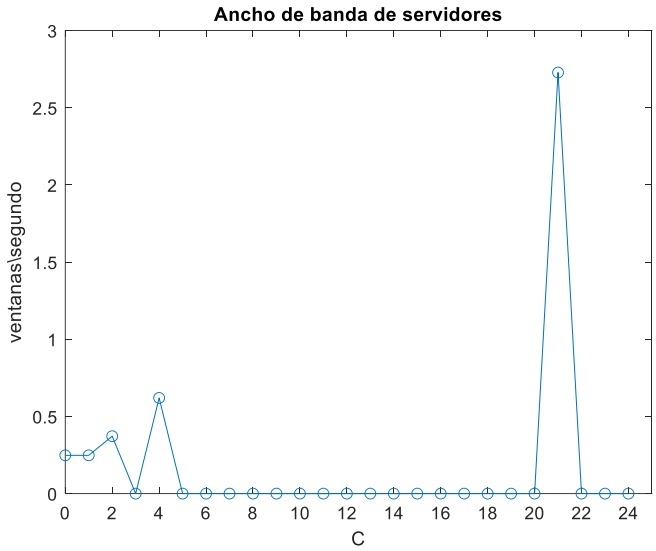


Figura 39. Ancho de banda proveniente de los servidores.

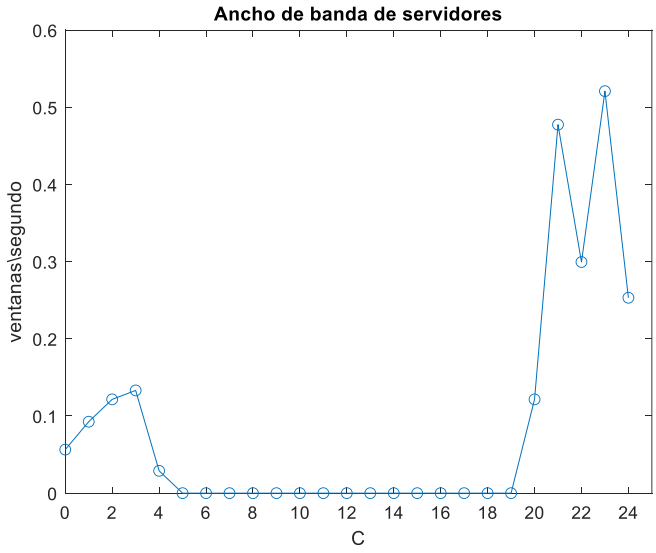


Figura 40. Ancho de banda proveniente de los servidores por iteración.

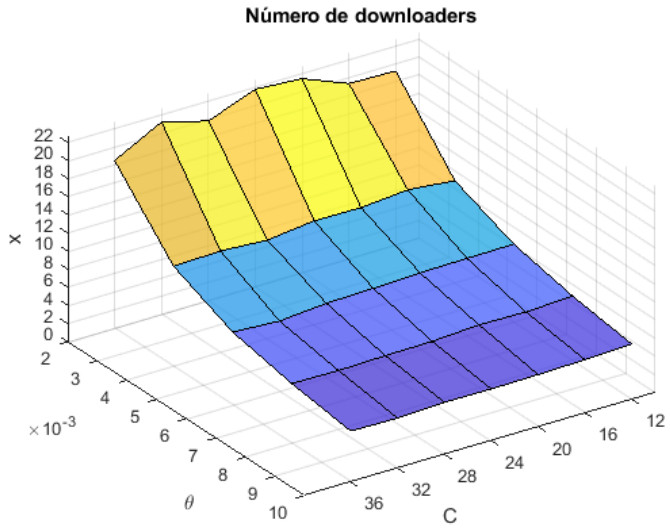


Figura 41. Downloaders promedio en el sistema livestreaming

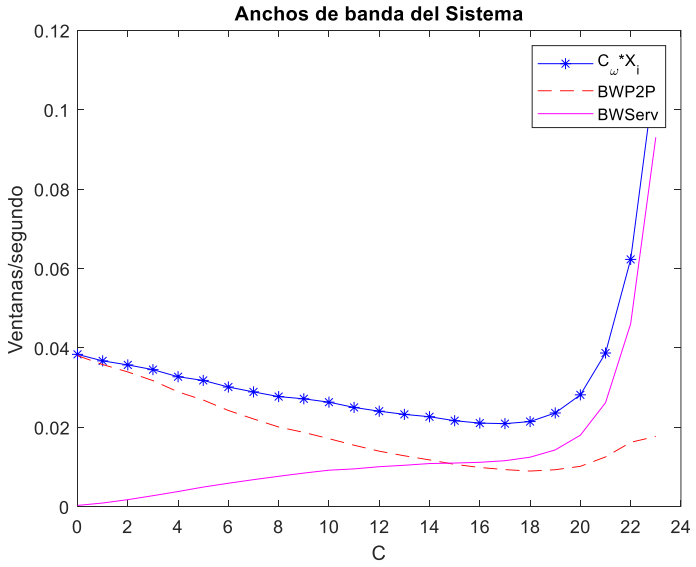
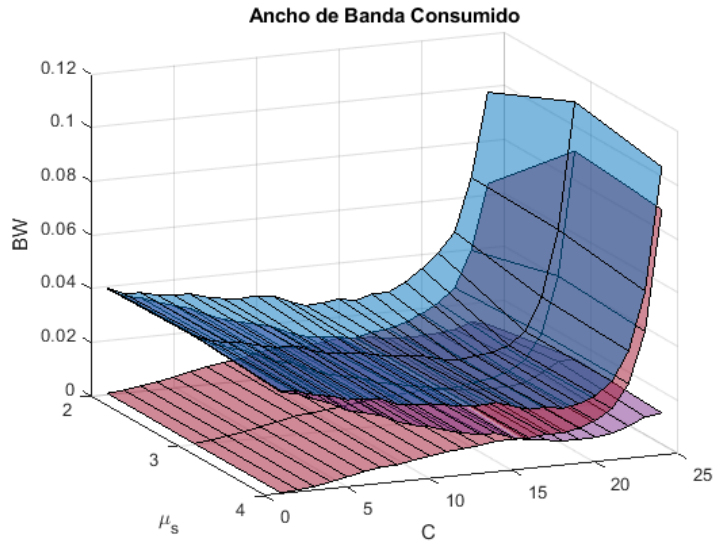
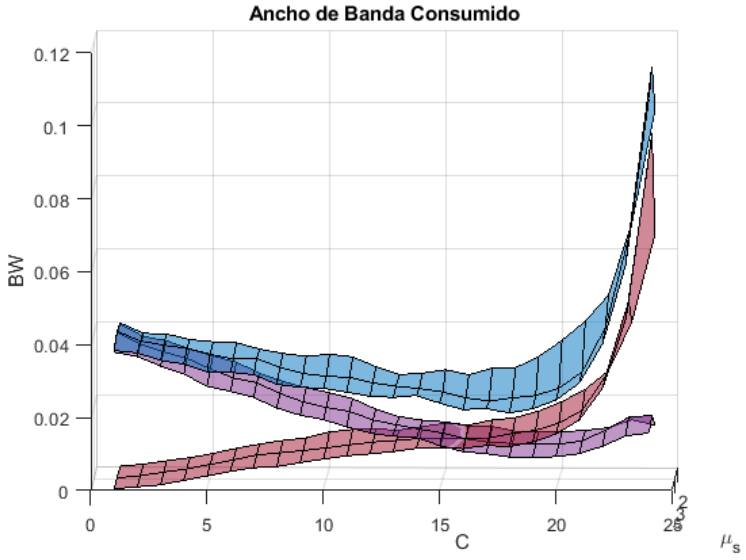


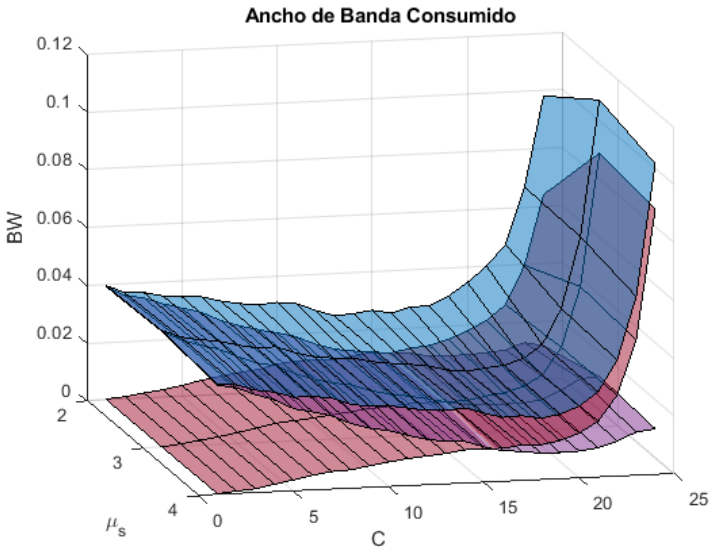
Figura 42. Ancho de Banda consumido en la descarga de video

Capítulo 7

Resultados






Ancho de Banda Consumido
BW
0.12
0.1
0.08
0.06
0.04
0.02
0
2
3
4
μ_s
0
5
10
15
20
25
C

Conclusiones

Con base en la problemática que se desarrolla en el presente proyecto y siguiendo los objetivos planteados se han realizado diversas actividades que permiten plantear una propuesta de solución asertiva. Se realizó el análisis de esquemas de asignación de recursos reportados en trabajos relacionados enfocados a servicios de video bajo demanda (*Video on Demand-VoD*) con el propósito de sentar bases para el desarrollo y evaluación de un esquema de asignación de recursos ad hoc para servicios de video en vivo.

Otra actividad fundamental fue el análisis del modelado de servicios de video bajo demanda mediante cadenas de Markov que permitió desarrollar una cadena de Markov que representa la dinámica de un sistema de transmisión de video en vivo sobre una red *P2P-CDN*.

La implementación de la solución por simulación de una cadena de Markov unidimensional permitió conocer en un ambiente simulado la transición en el estado de una cadena de Markov una vez que ocurre un evento del tipo discreto dentro del sistema modelado a través de la cadena. Por otro lado, esta implementación permitió conocer el *IDE* Matlab, su sintaxis y las variables resultantes que entrega una vez que se simulan eventos discretos.

Debido a que la implementación de la solución por simulación de la cadena unidimensional se realizó sin grandes problemas referentes a la lógica de programación, la sintaxis y las herramientas ofrecidas por el *IDE* Matlab se ha elegido a este para realizar la implementación de la solución por simulación de la cadena desarrollada en Proyecto Terminal I que representa a un sistema de servicio de video en vivo.

Una vez realizadas las actividades descritas anteriormente se espera que la implementación de la solución por simulación correspondiente a la cadena de Markov para servicios de video en vivo a lo largo de Proyecto Terminal II no represente complicaciones salvo por la lógica y sintaxis de programación dentro del *IDE* seleccionado.

Finalmente, se espera que durante el desarrollo de Proyecto Terminal II el comportamiento del sistema de servicios de video en vivo analizado y obtenido en Proyecto Terminal I sea el mismo o muy aproximado al que se obtendrá con la implementación de la solución por simulación de la cadena de Markov para servicios de video en vivo.

Por otro lado, se espera que el esquema de asignación de recursos a proponer en el presente proyecto muestre un desempeño eficiente frente a los servicios de video en vivo, es decir, el esquema de asignación de recursos debe ser capaz

de producir una cooperación efectiva entre los *peers* de la red *P2P* y por lo tanto disminuir las peticiones de atención a la red *CDN*.

Referencias

[1] T. Cooper, «BROADBANDNOW,» 28 Octubre 2020. [En línea]. Available: https://broadbandnow.com/report/cable-vs-satellite-vs-iptv-vs-ott-streaming/#:~:text=Essentially%2C%20IPTV%20is%20a%20formally,available%20to%20consumers%20from%20ISPs.&text=Rather%20than%20streaming%20content%20directly,to%20display%20on%20your%20TV.. [Último acceso: 21 Mayo 2021].

[2] S. M. Y. Seyyedi y B. Akbari, «Hybrid CDN-P2P Architectures for Live Video,» de *International Symposium on Computer Networks and Distributed Systems (CNDS)*, Tehran, 2011.

[3] A. Mansy y M. Ammar, «Analysis of Adaptive Streaming for HybridCDN/P2P Live Video Systems,» IEEE, Atlanta, 2011.

[4] R. Trestian, I.-S. Comsa y M. Fatih, «Seamless Multimedia Delivery Within aHeterogeneous Wireless NetworksEnvironment: Are We There Yet?,» *IEEE COMMUNICATIONS SURVEYS & TUTORIALS,* vol. 20, nº 2, pp. 945-977, 2018.

[5] N. Torres Cruz, M. E. Rivero Angeles, G. Rubino, R. Menchaca Mendez y R. Menchaca Mendez, «A Window-Based, Server-Assisted P2P Network forVoD Services with QoE Guarantees,» *Hindawi,* vol. 2017, nº 2084684, pp. 1-18, 2017.

[6] N. Torres Cruz, M. E. Rivero-Angeles, G. Rubino, R. Menchaca Mendez, R. Mechaca Mendez y D. Ramirez, «A comprenhensive analytical framework VoD Services in hybrid CDN-P2P,» *Elsevier ,* vol. 102643, nº 161, pp. 1-17, 2020.

[7] E. D. Terrones Celis, Artist, *Modelado y evaluación de servicios de video sobre redes P2P en ambientes móviles 5G.* [Art]. Centro de Investigación en Computación, 2020.

[8] T. T. Thu Ha, J. Kim y J. Nam, «Design and Deployment of Low-Delay Hybrid CDN–P2PArchitecture for Live Video Streaming Over the Web,» *Springer Science+Business Media,* vol. 10, nº 94, p. 13, 2015.

[9] M. Hassan, C. K. Neng y L. C. Suan, «Performance Analysis of Video Streaming on different HybridCDN & P2P Infrastructure,» MIMOS Berhad, Malaysia, 2021.

[10] S. Nacakli y A. Tekalp, «Controlling P2P-CDN Live Streaming Services at SDN-enabled Multi-Access Edge Datacenters,» *Fellow,IEEE,* vol. 10, nº Y, pp. 1-12, 2020.

[11] J. Sun, Y. Zhou, Y. Duan y Z. Guo, «A Low-latency Peer-to-Peer Live and VODStreaming System Based on Scalable Video Coding,» Beijing, IEEE, 2014, p. 319.

[12] N. Torres Cruz, *Esquemas de asignación de recursos para servicios de video en redes heterogéneas 5G,* México: IPN, 2019.

[13] V. Pichardo Herrera, *Evaluación de esquemas de asignación de recursos,* México: IPN, 2021.

[14] C. Zhao, J. Zhao, X. Lin y C. Wu, «Capacity of P2P on-demand streaming with simple, robust and decentralized control,» IEEE, Turin, 2013.

[15] BlackBox, «BlackBox,» 20 Julio 2018. [En línea]. Available: https://www.blackbox.com.mx/mx-mx/page/40830/Recursos/Technical/black-box-explica/Multimedia/Compresion-de-video-H264. [Último acceso: 22 Mayo 2021].

[16] R. Catro, «WIKIVERSUS,» 25 Julio 2020. [En línea]. Available: https://www.wikiversus.com/fotografia-y-video/codecs-h-265-vs-vp9/. [Último acceso: 22 Mayo 2021].

[17] C. Cano y M. Raffo, «Arqeuitectura de alta frecuencia de un filtro de escalabilidad para sobremuestreo de imágenes en factor 2 sobre una FPGA,» 2013.

[18] J. Alvaro, «Formatos Multimedia: Codificación y Empaquetado,» 2014. [En línea]. Available: https://edu.fauno.org/R.multimedia.formatos.html. [Último acceso: 22 Mayo 2021].

[19] UMA, «Herramientas web para la enseñanza de protocolos de comunicación,» evirtual, [En línea]. Available: https://neo.lcc.uma.es/evirtual/cdd/tutorial/presentacion/mpeg.html. [Último acceso: 5 Junio 2021].

[20] B. Jedari, G. Premsankar, G. Illahi, M. DiFrancesco, A. Mehrabi y A. Ylä-Jääski, «Video Caching, Analytics, and Delivery at the,» *IEEE COMMUNICATIONS SURVEYS & TUTORIALS,* vol. 23, nº 1, pp. 431-471, 2021.

[21] V. Rodríguez, «Scenikus blog,» 21 Abril 2021. [En línea]. Available: https://blog.scenikus.com/diferencia-entre-live-streaming-y-vod/. [Último acceso: 23 Mayo 2022].

[22] M. Willbert, «Live Streaming vs. VOD: Comparing the 10 Best Video Platforms That Do Both,» datacast, 20 Enero 2022. [En línea]. Available: https://www.dacast.com/es/blog/live-streaming-vs-vod/. [Último acceso: 23 Mayo 2022].

[23] D. Matus, «digitaltrends,» 18 Noviembre 2020. [En línea]. Available: https://es.digitaltrends.com/entretenimiento/tv-streaming-comparativa-servicios/. [Último acceso: 22 Mayo 2021].

[24] M. Wilbert, «HLS vs. MPEG-DASH: A Live Streaming Protocol Comparison for 2021,» datacast, 3 Mayo 2021. [En línea]. Available: https://www.dacast.com/blog/mpeg-dash-vs-hls-what-you-should-know/. [Último acceso: 5 Mayo 2021].

[25] W. J. Stewart, Probability, Markov Chains, Queues, and Simulation, New Jersey: Princeton University Press, 2009.