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.

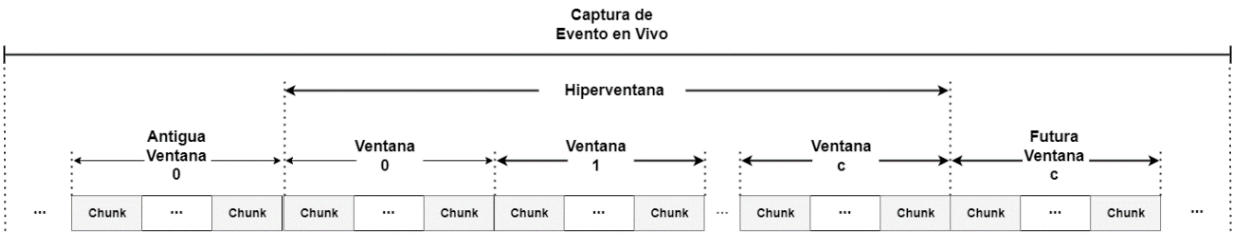


Figura 1. Estructura de una hiperventana

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 1. 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:

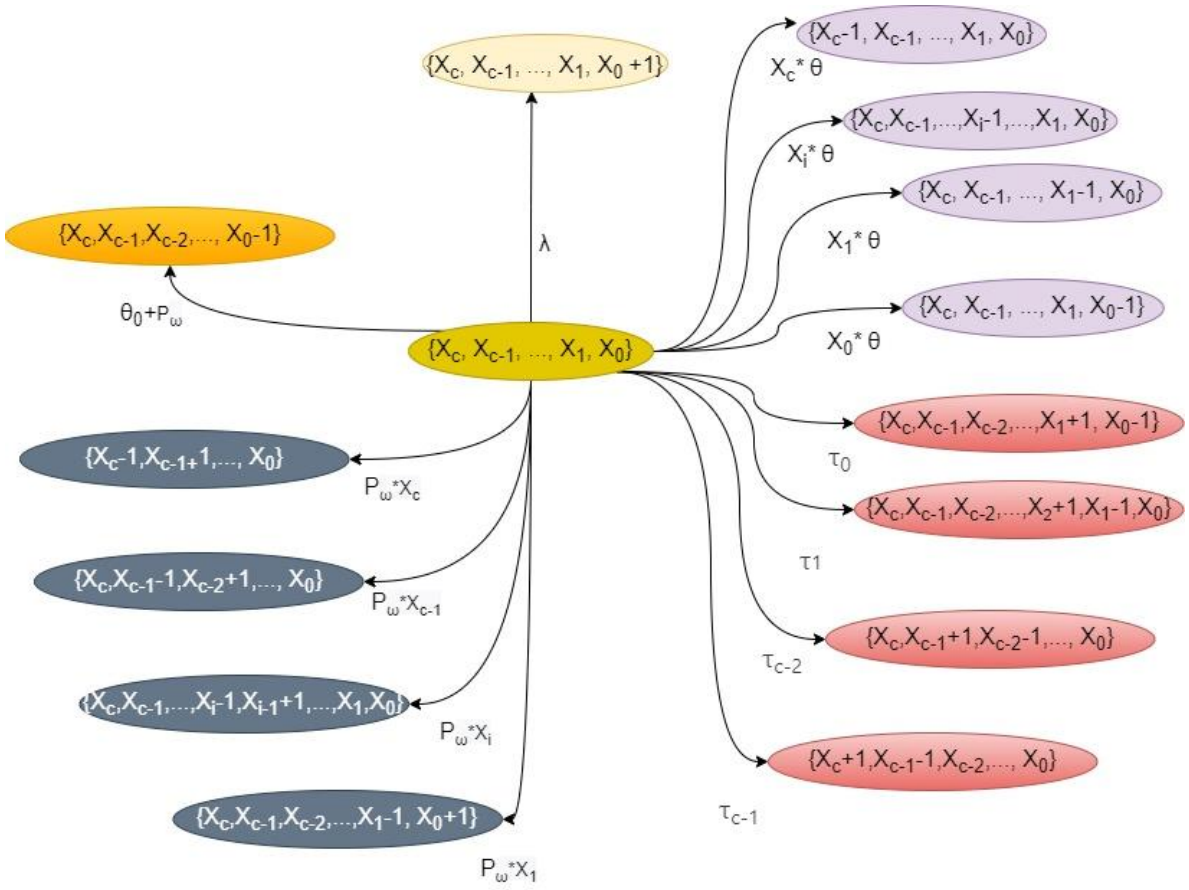


Figura 2. Cadena de Markov de un sistema de transmisión de video en vivo

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.

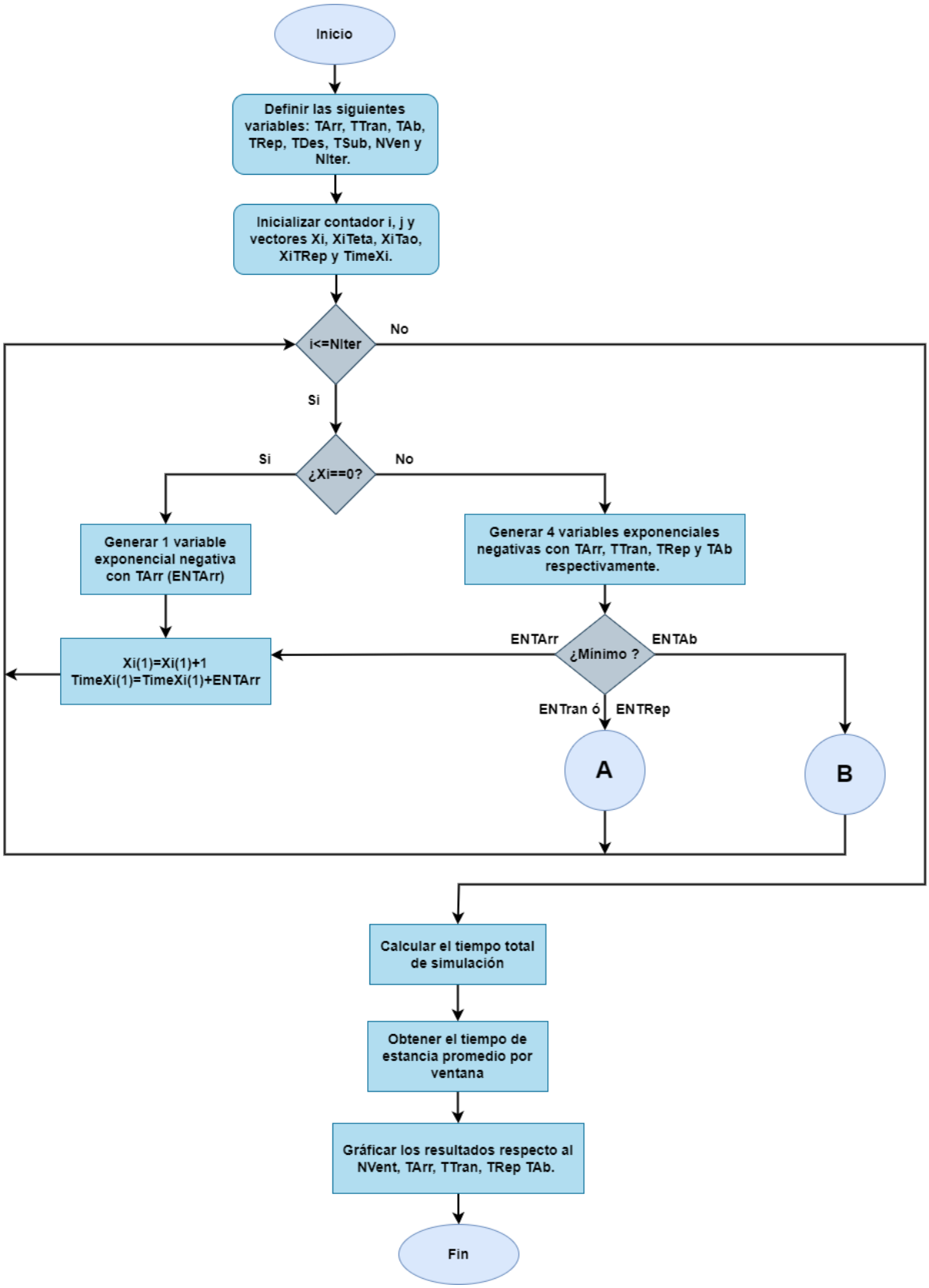

Inicio
Definir las siguientes variables: TArr, TTran, TAb, TRep, TDes, TSub, NVen y NIter.
Inicializar contador i, j y vectores Xi, XiTeta, XiTao, XiTRep y TimeXi.
i<=NIter
No
Si
¿Xi==0?
Si
No
Generar 1 variable exponencial negativa con TArr (ENTArr)
Generar 4 variables exponenciales negativas con TArr, TTran, TRep y TAb respectivamente.
Xi(1)=Xi(1)+1
TimeXi(1)=TimeXi(1)+ENTArr
ENTArr
¿Mínimo ?
ENTAb
ENTran ó
ENTRep
A
B
Calcular el tiempo total de simulación
Obtener el tiempo de estancia promedio por ventana
Gráficar los resultados respecto al NVent, TArr, TTran, TRep TAb.
Fin

Figura 3. 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.

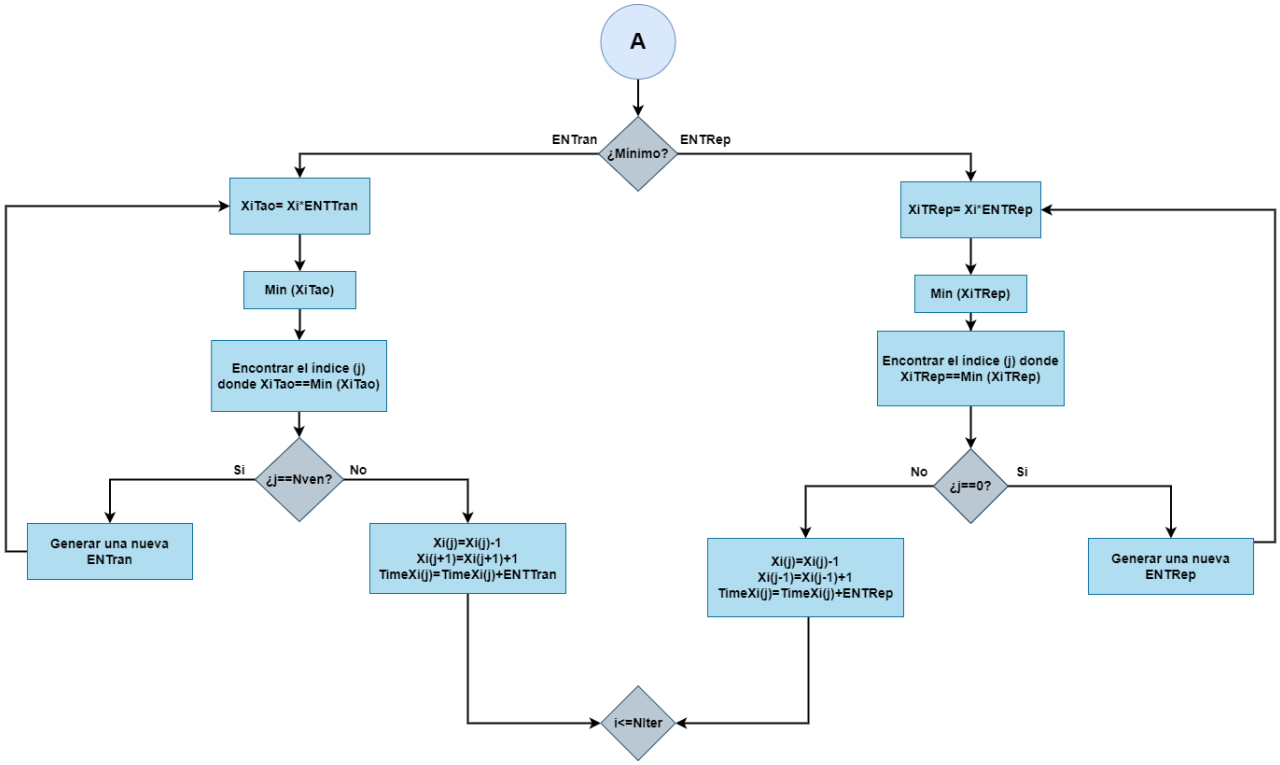


Figura 4. Diagrama para mín=ENTran ó EnTRep

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.

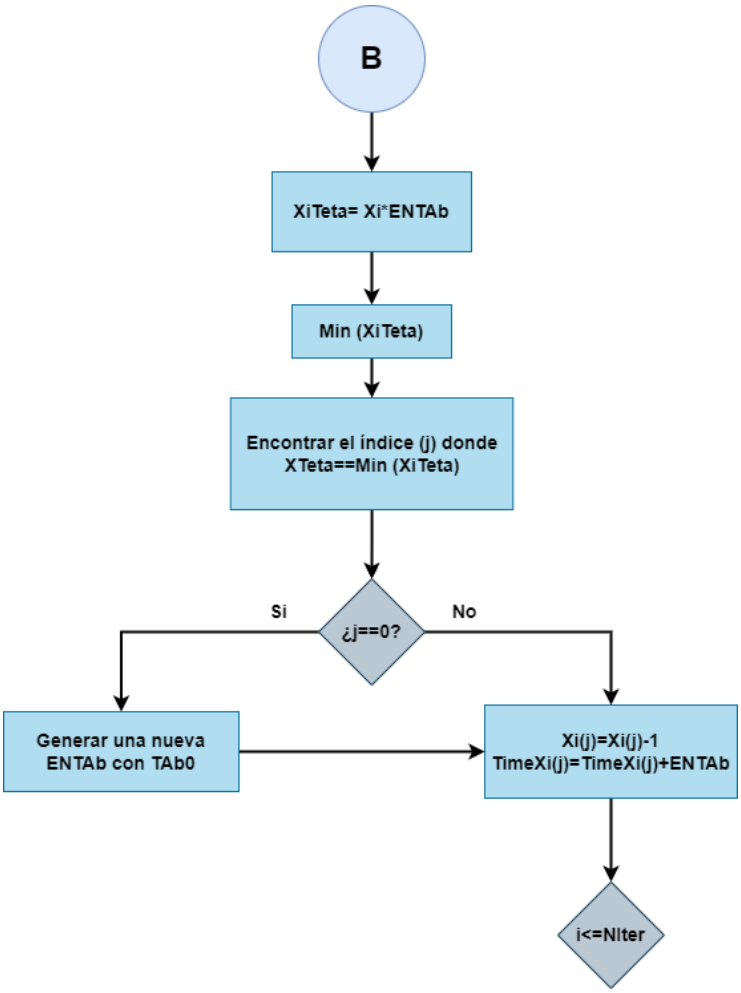


Figura 5. Diagrama para mín=ENTAb

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 en el 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 pueda descargar de manera satisfactoria el video en vivo y así poder establecer una expresión que permite 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 y visualizar 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 también 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 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 consuma el contenido almacenado en su *buffer*, detenga la descarga del video y presente congelamiento en la visualización del contenido.

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*.

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 que descargan el archivo de video mientras lo reproducen. 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 a otros *peers* que se encuentren en ventanas inferiores dentro del sistema. Este hecho permite que el servidor no se sature y no se le demande un ancho de banda mayor, por lo tanto, el servidor únicamente se centra en abastecer a los *peers* que están conectados directamente a él y con esto se crea una cadena de transferencia con una buena transmisión en el video en vivo. Sin embargo, el sistema debe estar preparado y tener reservado un espacio del ancho de banda proveniente de la red *CDN* para los posibles eventos que puedan ocurrir con los *peers*. Ya sea que los recursos provenientes de los *peers* sean insuficientes o un *peer* pierda la conexión a la transmisión de video.

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 $j+i$ a los *peers* en ventanas j, $0 \leq j < j+i$, 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 [7].

En la Figura 24, se observa la forma en que son distribuidos los recursos dentro del sistema de servicio de video en vivo y las expresiones que permiten realizar dicha asignación.

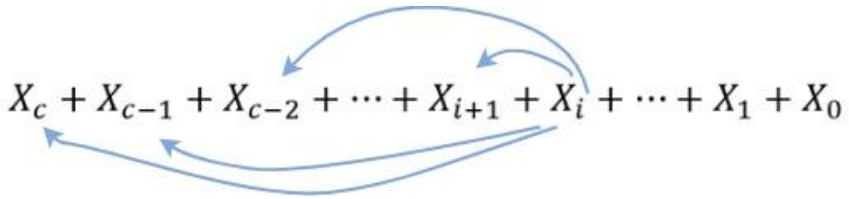


Figura 6. Obtención de recursos para la población en la ventana i

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 j tal que $i + 1 \leq j \leq c$, como se muestra en la siguiente expresión:

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

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

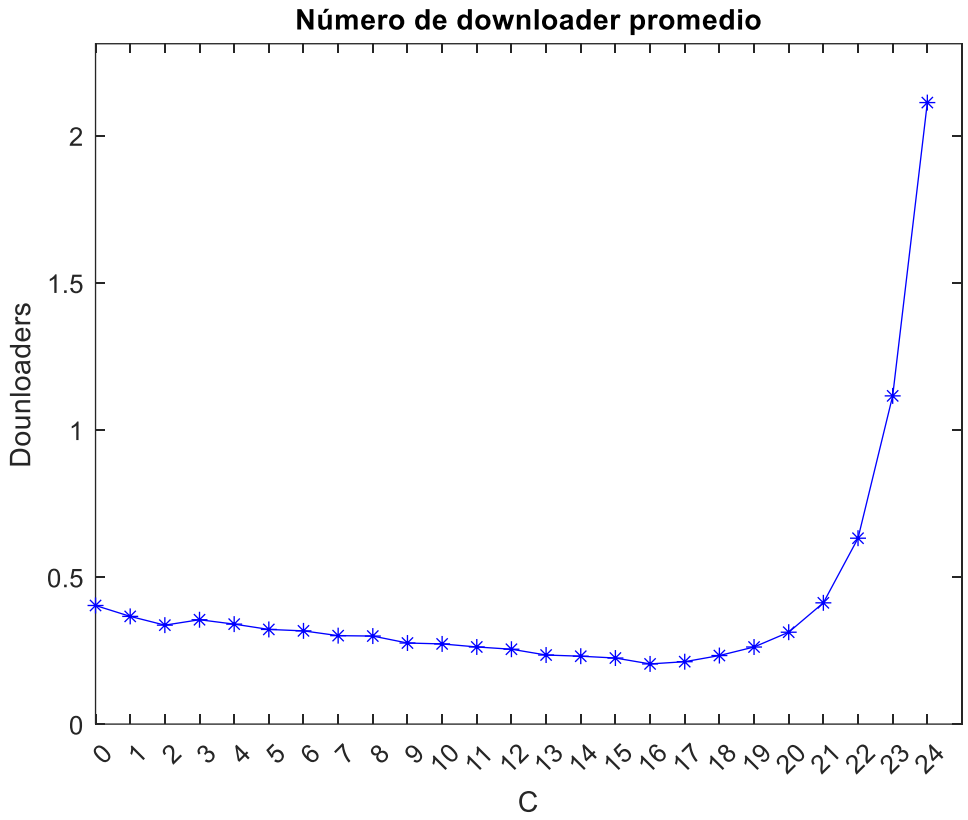
$$\left((\mu_\omega * X_{c-1}) \left(\frac{X_i}{X_0 + X_1 + \cdots + X_{c-2}} \right) + \mu_s \left(\frac{X_i}{X_0 + X_1 + \cdots + X_c} \right) \right) +$$

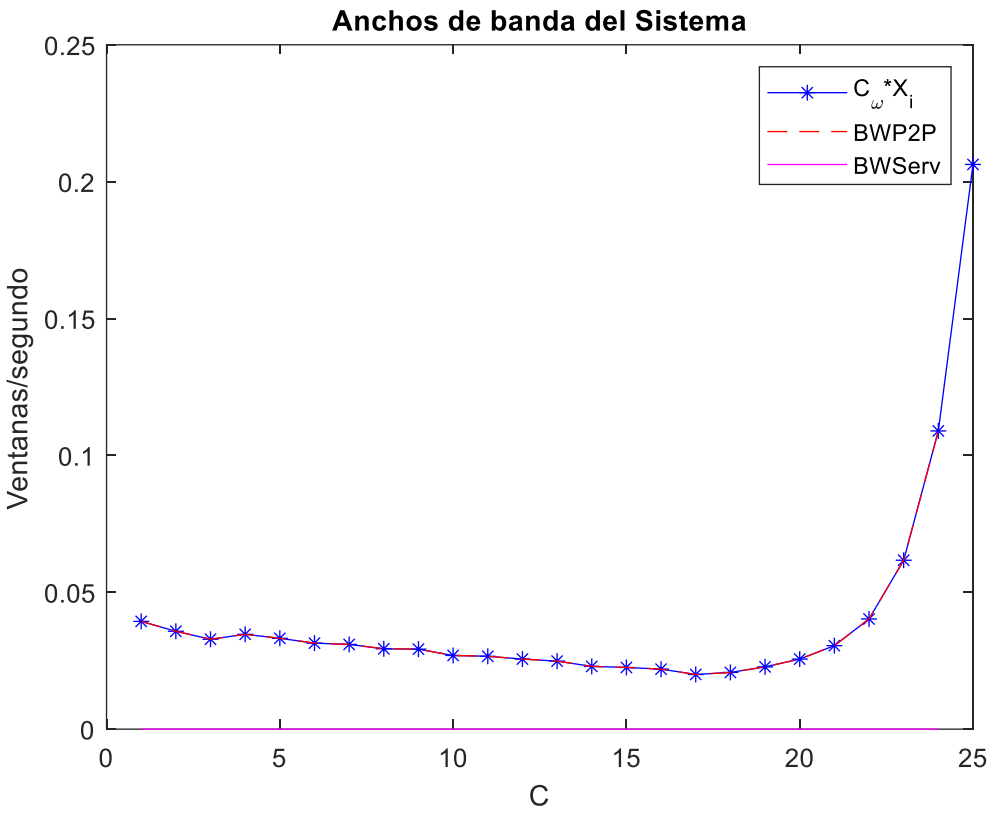
$$\left((\mu_\omega * X_c) \left(\frac{X_i}{X_0 + X_1 + \cdots + X_{c-1}} \right) + \mu_s \left(\frac{X_i}{X_0 + X_1 + \cdots + X_c} \right) \right)$$

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:

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

Resultados obtenidos




Anchos de banda del Sistema
0.25
0.2
0.15
0.1
0.05
0
Ventanas/segundo
0
5
10
15
20
25
C
C_ω*X_i
BWP2P
BWServ