Un Planificador de Canales Lógicos para un Servidor de VoD en Internet

Javier Balladini, Leandro Souza, Remo Suppi, y E. Luque *
Departamento de Arquitectura de Computadores y Sistemas Operativos
Universitat Autònoma de Barcelona
Bellaterra, 08193, España
{javier,leandro}@aomail.uab.es, {remo.suppi, emilio.luque}@uab.es

Abstract

Most of the Video on Demand (VoD) systems were designed to work in dedicated networks. However, there are some approaches that provide VoD service in nondedicated and best effort networks. In such case, they adapt the media's quality according to the available network bandwidth. Our research focus on VoD systems with high quality service over nondedicated networks. Currently, we have designed and developed a VoD server that includes a Network Traffic Scheduler (NTS) that is in charge to manage and transmit the data by the network, and a Logical Channels Scheduler (LCS) that schedule the videos' delivery to the different clients. The NTS component is in charge to inform the LCS of the communication state with each client in order to let LCS make the videos' delivery schedule with knowledge of the network state. The present work describes the LCS component and the results of the integration with the NTS, demonstrating the well function and viability to work over networks of variable conditions such as the Internet.

Keywords: VoD server, logical channels scheduler, media's delivery, network of variable conditions

Resumen

La mayoría de los sistemas de Vídeo bajo Demanda (VoD, Video on Demand) fueron diseñados para trabajar en redes dedicadas. Sin embargo, hay algunos sistemas que proveen servicio de VoD en redes de mejor esfuerzo y no dedicadas, pero ellos adaptan la calidad de la media de acuerdo al ancho de banda disponible de la red. Nuestras actividades de investigación se enfocan en sistemas de VoD con alta calidad de servicio en redes no dedicadas. Actualmente, hemos diseñado y desarrollado un servidor de VoD, que incluye un Planificador del Tráfico de Red (NTS) encargado de la gestión y transmisión de los datos por la red, y un Planificador de Canales Lógicos (LCS) que planifica la entrega de los vídeos a los diversos clientes. El componente NTS es el encargado de informar al LCS sobre el estado de la comunicación con cada uno de los clientes para que éste pueda planificar la entrega de los vídeos con conocimiento del estado de la red. El presente trabajo describe el componente LCS y los resultados de su integración con el NTS, demostrando el correcto funcionamiento y viabilidad para trabajar en una red de condiciones variables como la red Internet.

Keywords: servidor de VoD, planificador de canales lógicos, entrega de vídeos, red de condiciones variables

*Este trabajo está soportado por el MEyC-España bajo el contrato TIN 2004-03388.

1. INTRODUCCIÓN

Un sistema de Vídeo bajo Demanda (VoD, Video on Demand) brinda un servicio de visualización de vídeos a los usuarios. El tipo de servicio más completo de VoD permite al usuario solicitar su vídeo preferido cuando lo desee y reproducirlo casi instantáneamente, pudiendo utilizar comandos interactivos (congelado de imagen, retroceso, avance, cámara lenta/rápida, etc.) tal como si estuviese visualizando el vídeo con un reproductor de VHS o DVD. A este tipo de servicio se lo conoce como T-VoD (True Video on Demand). Un sistema de VoD presenta una elevada complejidad y altos requerimientos de recursos, especialmente de la red de comunicación. No obstante, el avance de la tecnología en el campo de las redes, hace posible que, en la actualidad, la red Internet pueda ser considerada para proporcionar servicios como los de VoD.

La mayoría de los sistemas de VoD fueron diseñados para trabajar en redes dedicadas. Sin embargo, hay algunos esquemas que proveen servicio de VoD en redes no dedicadas y de mejor esfuerzo, pero ellos adaptan la calidad de la media de acuerdo al ancho de banda disponible.

Nuestras lineas de investigación están orientadas al desarrollo de una arquitectura para sistemas de VoD a gran escala (LVoD, Large Video on Demand), que provea un servicio T-VoD de alta calidad, manteniendo la calidad de la media durante todo el tiempo de servicio, sobre redes no dedicadas y que tienen un modelo de servicio de mejor esfuerzo, como lo es la red Internet.

En la Internet, la calidad de servicio y los recursos no están garantizados. Como la red no es dedicada, el ancho de banda varía dependiendo del tráfico que esté circulando en cada momento. El retardo también varía dependiendo entre otras cosas del camino que recorra la información por la red. Además, hay pérdida de paquetes ya sea por razones físicas o por descartes. En el primer caso, se incluye el ruido que afecta las señales o las fallas de los dispositivos activos de comunicación (enlaces, encaminadores -routers-, etc), mientras que el descarte de paquetes es debido a estados de congestionamiento.

Un sistema de VoD que hace uso de técnicas de transmisión por multitransmisión (multicast), permite compartir recursos entre un número elevado de clientes. Sin embargo, debido a que el tipo de comunicación por multitransmisión está implementado en una reducida parte de la red Internet, se ha optado por utilizar la transmisión por canales simples (unicast), por ser el único tipo de comunicación soportado por la totalidad de esta red.

La figura 1 muestra un diagrama de alto nivel de la arquitectura del sistema de VoD propuesto. En este diagrama puede observarse la arquitectura del Cliente y del Servidor de VoD o VPS (Video Proxy Server), y la red a través de la cual se efectúan las comunicaciones. El Cliente realiza una petición de un determinado vídeo al componente Control de Admisión (AC, Admission Control) del servidor, quien toma la decisión de aceptar o rechazar la petición. Esta decisión se basa en la información recibida desde el Gestor de Recursos (RM, Resource Manager) sobre la disponibilidad o no de los recursos necesarios para su atención. El componente AC asegura que la aceptación de un nuevo cliente al sistema no compromete el buen funcionamiento del mismo. Cuando el AC acepta una nueva petición, se crea un nuevo canal lógico o *stream* (flujo de datos) para atender el requerimiento.

El Gestor de Recursos (RM, Resource Manager) es el componente que gestiona la información de los recursos del sistema relativos a CPU, memoria, disco, y red. El RM contabiliza los recursos consumidos y los libres, y permite reservar recursos para atender a las peticiones de los clientes.

El Planificador de Canales Lógicos (LCS, Logical Channels Scheduler) es el encargado de decidir qué canal lógico es servido en cada momento para garantizar la QoS. Distribuye el ancho de banda de salida de un servidor entre los distintos *streams* de tal manera que los clientes reciban la media con la calidad adecuada.

El Planificador del Tráfico de Red (NTS, Network Traffic Scheduler) brinda servicio al LCS,

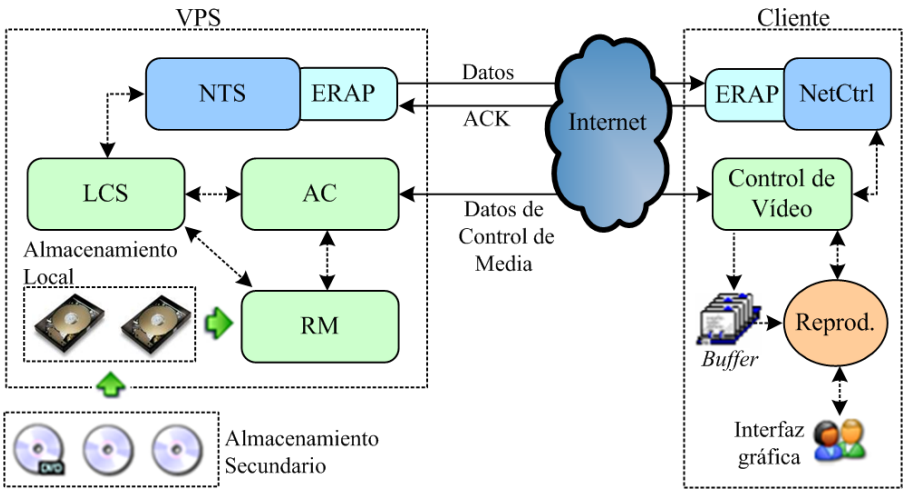


Figura 1: Arquitectura del VPS y del Cliente

proporcionando un transporte confiable de los datos que el LCS desea enviar a los clientes. El NTS se encarga de todos los aspectos de la red, incluyendo el control de la congestión de la misma, y de reportar al LCS sobre el estado de la conexión del servidor con cada uno de los clientes. De esta manera, el LCS puede planificar correctamente la distribución del ancho de banda entre los diferentes clientes.

El NTS que hemos desarrollado y presentado en [1] utiliza el transporte proporcionado por el protocolo UDP (User Datagram Protocol), e incluye un algoritmo de control de congestión denominado ERAP (Enhanced Rate Adaptation Protocol) para que el VPS pueda ser utilizado en la Internet, con garantías de calidad de servicio (QoS). El NTS no solo coopera para el buen funcionamiento del VPS, sino que también coopera para el buen funcionamiento de la red Internet, ya que el protocolo ERAP es TCP-Friendly. Esto quiere decir que no consume más ancho de banda (en un estado estable) que el protocolo TCP bajo circunstancias similares [3].

ERAP esta basado en la especificación del protocolo de control de congestión denominado RAP (Rate Adaptation Protocol) [14, 13]. ERAP mejora el protocolo RAP, no desde el punto de vista del comportamiento en el ajuste de la tasa de transmisión (que es muy similar), sino desde el punto de vista del consumo de recursos del servidor y la red. Nuestro protocolo elimina las limitaciones del protocolo RAP, dando al servidor de vídeo más flexibilidad y oportunidad para atender las peticiones de los clientes. La administración de eventos optimizada, la reducción del tamaño de los encabezados, la recepción centralizada de paquetes de reconocimiento (ACK, acknowledgement), y el envío centralizado de paquetes de datos, han sido las estrategias para obtener tal disminución del consumo de recursos.

Por otro lado, el Cliente dispone de todos los componentes básicos para reproducir vídeo, que es el Reproductor, la Interfaz Gráfica, el *Buffer* donde almacena el vídeo recibido, y el Control de Vídeo que es el componente encargado de comunicarse con el servidor para transmitirle peticiones de vídeos o la ejecución de comandos interactivos. A este cliente básico se le ha agregado el componente NetCtrl que mediante el protocolo ERAP recibe el vídeo enviado por el servidor.

1.1. Trabajos relacionados

En esta sección se presenta una visión general de los trabajos realizados en el área de calidad de servicio y planificación al nivel de aplicación y de red en los sistemas de VoD.

Se han presentado varias propuestas para aplicaciones que envían media prealmacenada, y en especial para media continua prealmacenada "semisoft". Las aplicaciones "semisoft" tienen muy poca tolerancia temporal inicial, y por lo tanto, solo una pequeña cantidad de información puede ser enviada por adelantado. Dentro de estos esquemas pueden citarse a [18], [17], y [7] . Sin embargo, no son apropiados para nuestros objetivos porque están diseñados para redes específicas donde los dispositivos activos de comunicación incluyen capacidades adicionales y particulares para manejar el tráfico. Otros esquemas tienen el problema de que la asignación de recursos de red es estática y es determinada al inicio de la sesión, tal es el caso de [9].

Nuestro grupo de investigación ha propuesto en [10, 11] el algoritmo CB_MDA, sin embargo está ideado para un entorno de red dedicado y con reserva de recursos. Otras propuestas están diseñadas para adaptarse a las condiciones variables de la red Internet, pero bajan la calidad del vídeo ante estados de congestión, tal es el caso de [12].

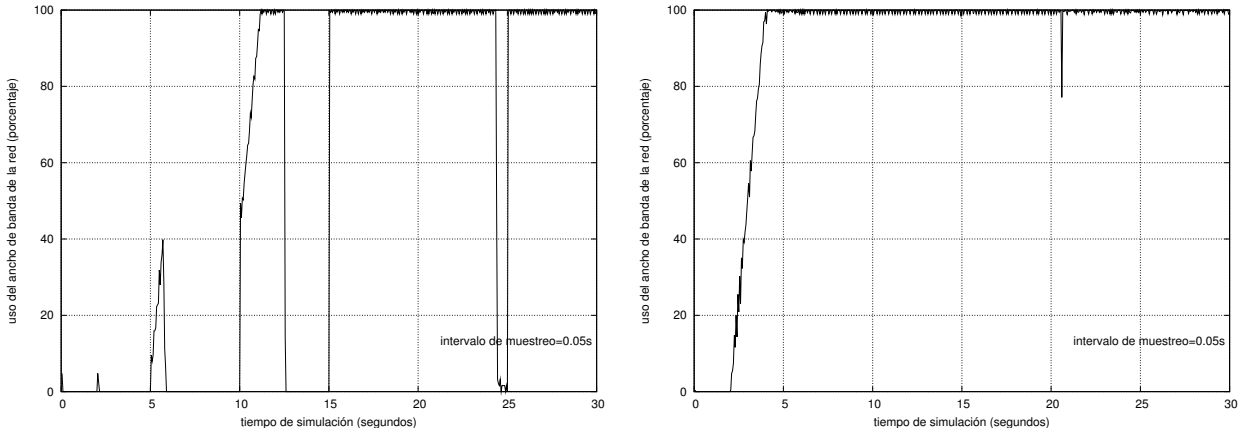
El control de congestión ha sido estudiado por muchos años, no obstante, los protocolos existentes no son muchos si nos restringimos a protocolos TCP-Friendly para transmisión multimedia en la red Internet. Jacob et al. [6] presenta un algoritmo de control de congestión similar a TCP excepto que este no hace retransmisiones. Cent et al. [2] propone el protocolo SCP (Streaming Control Protocol) para transmisión en tiempo real de *streams* (flujos de datos) multimedia continuos por la Internet. Dorgham Sisalem et al. presenta el protocolo LDA (Loss-Delay Adjustment Algorithm) en [15] y su variante LDA+ en [16]. Sally Floyd et al. expone el protocolo TFRC (TCP-Friendly Rate Control) en [4], y mas tarde, M. Handley et al. describe este protocolo en el RFC 3448 [5]. Reza Rejaie et al. presenta en [14, 13] el protocolo RAP (Rate Adaptation Protocol) que incluye un control de congestión de extremo a extremo, basado en tasa, para la transmisión en tiempo real de *streams* por la red Internet. Nuestro protocolo ERAP está basado en este último protocolo, el cual fue re-diseñado para aumentar sus prestaciones en un servidor con alta carga de comunicaciones.

2. LCS: PLANIFICADOR DE CANALES LÓGICOS

El objetivo del Planificador de Canales Lógicos (LCS, Logical Channels Scheduler) es distribuir el ancho de banda total de salida del servidor entre los diferentes *streams* (flujos de datos) garantizando la QoS. El LCS que hemos diseñado es del tipo "anticipación" y mantiene la calidad del vídeo entregado al cliente a lo largo de toda su reproducción. Un LCS del tipo "anticipación" envía media por adelantado que el cliente consumirá mas tarde. De esta manera, es posible tomar ventaja de períodos en que el ancho de banda disponible de la red es elevado, para enviar por adelantado la media que el cliente consumirá mas tarde, y así poder tolerar momentos en que la red o el servidor estén sobrecargados [18].

El tiempo se divide en unidades llamadas *Slots*, y dentro de cada *Slot* el LCS debe tomar dos decisiones: (1) ¿Cuales serán los *streams* seleccionados para ser servidos durante este *Slot*?, (2) ¿Cuánta media se enviará de cada *stream* elegido?. La toma de estas dos decisiones es la tarea principal que efectúa el algoritmo de planificación del LCS.

A continuación se presenta el análisis de la división del tiempo en *Slots*, y luego se describe el algoritmo de planificación del LCS que hemos desarrollado. El algoritmo de planificación que aquí se presenta, toma los conceptos básicos del algoritmo CB_MDA [10, 11] diseñado por nuestro grupo de investigación para un entorno de red dedicado y con reserva de recursos. El agregado de nuevas características permite su integración con el componente NTS de la arquitectura, que le provee los servicios de transmisión de datos a los clientes e información del estado de esas transmisiones.



(a) Slot de 5 segundos (b) Slot de 0,1 segundo

Figura 2: Comportamiento del LCS con distintos tamaños de *Slot*

2.1. División del tiempo en *Slots*

El LCS divide el tiempo en unidades denominadas *Slots*, y efectúa una planificación para cada una de estas unidades de tiempo. Por ejemplo, si el tamaño del *Slot* es de 1 segundo, el LCS realizará la planificación del *Slot* 1, y luego de transcurrir un nuevo segundo, realizará la planificación del *Slot* 2, y así sucesivamente.

El tamaño del *Slot* está estrechamente relacionado a la precisión de las planificaciones. Se han realizado algunas pruebas para observar cuan precisas son las planificaciones para diferentes tamaños de *Slot*. Las figuras 2 (a) y 2 (b) muestran los gráficos resultantes de las pruebas simuladas para tamaños de *Slot* de 5 y 0,1 segundos, donde se observa el porcentaje de uso del ancho de banda de salida del servidor a lo largo del tiempo.

Como ejemplo, observemos el gráfico de la figura 2 (a), con un tamaño de *Slot* de 5 segundos. En el tiempo 10, el LCS decide a qué clientes enviar media, y cuánta media enviarles. Como puede observarse, desde el segundo 11,5 aproximadamente, la red no es utilizada. Esto se debe a que la información con la cual el LCS ha tomado la decisión, era correcta para el tiempo 10, pero evidentemente no lo fue para todo el tiempo de duración del *Slot*.

El LCS toma información en un momento dado, y con esa información planifica todo el *Slot*, por lo tanto, cuanto más pequeño sea el tamaño del *Slot*, más precisa será la planificación. La forma de verificar la precisión de la planificación es observando el aprovechamiento del ancho de banda. Esto quiere decir que cuanto más ancho de banda se utilice, más precisa es la planificación. Sin embargo, hay que buscar un equilibrio, ya que cuanto menor sea el tamaño del *Slot*, mayor será la cantidad de planificaciones realizadas, y por lo tanto, mayor será el consumo de recursos de CPU.

Los resultados han determinado que realizar 10 planificaciones por segundo es suficiente para obtener una elevada precisión en las planificaciones con un uso de CPU razonable.

2.2. Algoritmo de planificación

En primer lugar, explicaremos el significado del *deadline*, un concepto fundamental que es la base de las decisiones del planificador. En segundo lugar, explicaremos cómo el algoritmo resuelve la planificación de los *streams* de vídeo.

Dado un cliente que está reproduciendo un vídeo de manera continua, el *deadline* se define como

la cantidad de tiempo que el cliente puede continuar reproduciendo, utilizando solamente la media que actualmente dispone en *buffer*.

Explicaremos, a continuación, el algoritmo planificador del LCS que hemos desarrollado. Cada cierto tiempo, determinado por la longitud del *Slot,* el algoritmo realiza una planificación correspondiente a un nuevo *Slot*. Inicialmente, se calcula, para este *Slot*, el número total de bytes que el LCS tiene disponible para enviar, y la cantidad máxima de bytes que el LCS puede transmitir a cada cliente en particular. Luego se entra en un proceso de selección de *streams* y envío de datos de esos *streams*. El proceso termina cuando la cantidad de datos enviados es igual a la cantidad máxima de bytes que el servidor tenía disponibles para enviar, o cuando no existen más *streams* que puedan ser seleccionados para ser servidos.

Ahora veamos cómo es el proceso de selección de *streams* que serán servidos durante este *Slot*. El algoritmo de planificación selecciona a aquellos clientes con menor *deadline*, dentro de los clientes con capacidad de recibir nuevos datos. El cliente con *deadline* mas pequeño es el que dispone de menos vídeo en *buffer*. De esta manera, nos aseguramos de servir primero a los clientes con más necesidades de media. Si al cliente ya se le ha enviado toda la media que es capaz de recibir en el tiempo de un *Slot*, deberá esperar hasta el siguiente *Slot* en el cual vuelva a tener disponibilidad para recibir más media.

Una vez efectuado el proceso de selección de *streams*, debe resolverse la siguiente cuestión: ¿Cuánta media se enviará de cada *stream* elegido?. La cantidad de datos que se envían es calculada según la ecuación 1.

$$sent = \text{mín}(emptyBlockSize, remainingCV, remainingSCV, MaxDataBlock) \qquad (1)$$

En ésta ecuación se definen las siguientes variables:

- *emptyBlockSize:* es el tamaño en bytes del siguiente bloque de datos vacío del *buffer* del cliente que se encuentra inmediatamente después del punto de reproducción.
- *remainingCV:* es la cantidad restantes de bytes que el LCS puede entregar al NTS para ser transmitidos a un cliente particular. Este valor depende de la capacidad de recepción del cliente y de los datos ya enviados durante el *Slot* actual.
- *remainingSCV:* indica la cantidad de bytes restantes que el LCS puede entregar al NTS durante este *Slot* para ser transmitidos a los clientes. Depende de la capacidad máxima de envío de datos del servidor por el enlace de salida, y de los datos ya enviados durante el *Slot* actual.
- *MaxDataBlock:* es la cantidad máxima de bytes que puede tener el bloque de datos que se va a enviar a un cliente cada vez que es seleccionado para ser servido.

La figura 3 presenta el pseudo-código del algoritmo planificador del LCS, donde los datos y funciones allí referenciadas son:

- *RM.getMaxSCV(slotLength):* retorna el valor de MaxSCV (Max Server Credit Value) que corresponde a un *Slot* de tamaño *slotLength* (expresado en segundos). Este valor, solicitado al componente Gestor de Recursos (RM), indica la cantidad máxima de bytes que el LCS puede entregar al NTS durante el *Slot* actual para ser transmitidos a los clientes. El valor de MaxSCV es calculado según la ecuación 2. En ésta ecuación, *NTS.getBandwidth()* es el ancho de banda de salida

```
NuevoSlot()
  remainingSCV = RM.getMaxSCV(slotLength)
  para cada stream S hacer {
    S.resetRemainingCV(slotLength)
  }
  hacer {
    encontrado = falso
    para cada stream S hacer {
      si (S.remainingCV > 0) {
        d = S.getDeadline
        si ((no encontrado) o (d < minDeadline)) {
          encontrado = verdadero
          minDeadline = d
          CS = S
        }
      }
    }
    si (encontrado) {
      enviado = CS.send(min(CS.emptyBlockSize, CS.remainingCV,
                            remainingSCV, MaxDataBlock))
      remainingSCV = remainingSCV - enviado
    }
  } mientras ((encontrado) y (remainingSCV > 0))
```

Figura 3: Pseudo-código del algoritmo planificador del LCS

del servidor (expresado en Mbps -Megabits por segundo-), el cual es obtenido mediante una prueba realizada al momento de iniciarse el servidor. La variable *slotLength* es es el tiempo de duración del *Slot*. La función *NTS.getEnqueuedMbits(slotLength)* retorna la totalidad de bytes que el NTS aún tiene encolados para transmitir durante los siguientes *slotLength* segundos.

$$MaxSCV = (BW * slotLength - enqueued) * 10^6/8$$

$$donde \quad \begin{array}{c} BW = NTS.getBandwidth() \\ enqueued = NTS.getEnqueuedMbits(slotLength) \end{array} \tag{2}$$

- *S.resetRemainingCV(slotLength)*: *remainingCV* es la variable mantenida para cada *stream*, donde se registra la cantidad restantes de bytes que el LCS puede entregar al NTS durante este *Slot*, para ser transmitidos al cliente. La función *resetRemainingCV* asigna la cantidad máxima de bytes que el LCS puede entregar al NTS durante este *Slot* cuya duración es especificada por el parámetro *slotLength*. El valor asignado a *remainingCV* se calcula basándose en la ecuación 3. En ésta ecuación, *sender.getMbps()* representa el ancho de banda actual del cliente expresado en Mbps. Esta información la provee el *sender*, que es un agente del NTS encargado de gestionar el envío de datos por la red de cada *stream*. La función *sender.getEnqueuedBytes()* (también provista por el agente *sender* del NTS) representa la cantidad de bytes que el LCS ha entregado al NTS para ser transmitidos al cliente, pero que aún no han salido a la red y, posiblemente, algunos otros bytes correspondientes a paquetes perdidos que el NTS necesite retransmitir.

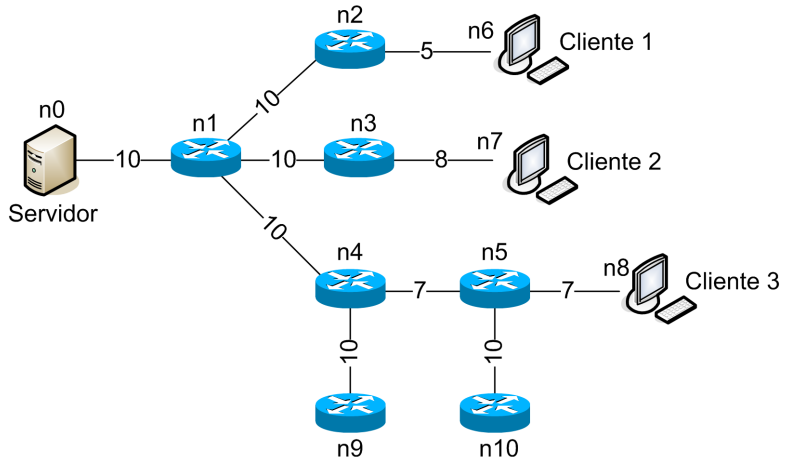


Figura 4: Topología para la simulación VPS vs. RAP y VPS vs. TCP

$$RemainingCV = \begin{cases} 0 & a \leq b \\ a - b & a > b \end{cases}$$

$$donde \qquad (3)$$

$$a = Slot * sender.getMbps() * 10^6/8$$
$$b = sender.getEnqueuedBytes()$$

- *S.getDeadline:* retorna el valor del *deadline* (expresado en segundos) correspondiente al *stream S*. Este valor es obtenido a partir de una estructura mantenida por el servidor que representa el *buffer* del vídeo del cliente, donde se identifican los bloques vacíos, los bloques con datos, y el punto de reproducción.

- *CS.send(length):* entrega al NTS los datos del vídeo correspondiente (de longitud *length*) al *stream* CS, para que este componente se encargue de transmitirlos al cliente.

3. RESULTADOS EXPERIMENTALES

Se ha desarrollado un prototipo del VPS propuesto para hacer experimentos reales y de simulación. En esta sección se presentan los resultados de algunas simulaciones, efectuadas con el simulador Network Simulator - NS2 [8], que muestran las ventajas que logra el VPS al integrar adecuadamente el control de flujo del nivel de aplicación con el control de flujo del nivel de la red.

En contraste, se presentan dos aplicaciones contra las cuales comparamos el VPS. Estas aplicaciones envían el vídeo a los clientes, una mediante el protocolo de transporte RAP y la otra mediante TCP, sin ninguna planificación de alto nivel.

En la figura 4 se presenta la topología de red simulada con los anchos de bandas (expresados en Mbps) correspondientes a cada enlace, y en el cuadro 1 se presenta el resto de los parámetros de la simulación.

Se tienen tres clientes, que comienzan a recibir el mismo vídeo en el mismo instante de tiempo. El servidor se encuentra en el nodo n0, el cliente 1 se encuentra en el nodo n6, el cliente 2 en el nodo n7, y el cliente 3 en el nodo n8.

Desde el segundo 10 hasta el segundo 15 se realiza un envío continuo de paquetes UDP, desde el nodo n9 al nodo n10. Este flujo no tiene ningún control de congestión. De esta manera, se apropia

Parámetro de la simulación	Descripción
Tamaño de paquetes	1040 bytes
Carga útil de los paquetes de datos	VPS-ERAP: 1004 bytes RAP: 988 bytes TCP: 996 bytes
Retardo de los enlaces	10 ms
Tipo de cola de mensajes	*DropTail*
Tamaño de la cola de mensajes del nodo n0	RAP y TCP: 1000 paquetes VPS: 10 paquetes
Tiempo de simulación	240 segundos

Cuadro 1: Parámetros para la simulación VPS vs. RAP y VPS vs. TCP

durante el tiempo de duración del flujo, de casi todo el ancho de banda del enlace comprendido entre los nodos n4 y n5.

A continuación explicaremos cómo interpretar los gráficos de "tiempo vs. *deadline*" de las simulaciones que son presentadas mas adelante.

El eje *x* del gráfico "tiempo vs. *deadline*" representa el tiempo (segundos) de reproducción del vídeo por parte del cliente, comenzando por el segundo 0. Los valores negativos de la escala del eje *x* representan el tiempo de *prefetch*. Es decir, la media que el cliente almacena en *buffer*, pero que no comienza a consumir hasta haber transcurridos cierto tiempo desde que comenzó a recibir el vídeo. En este caso, se realiza un *prefetch* de 10 segundos, comprendido desde el tiempo -10 al tiempo 0.

El eje *y* de este gráfico representa el *deadline*. Los valores negativos de la escala representan el tiempo que el cliente ha detenido la reproducción del vídeo por falta de media en *buffer*.

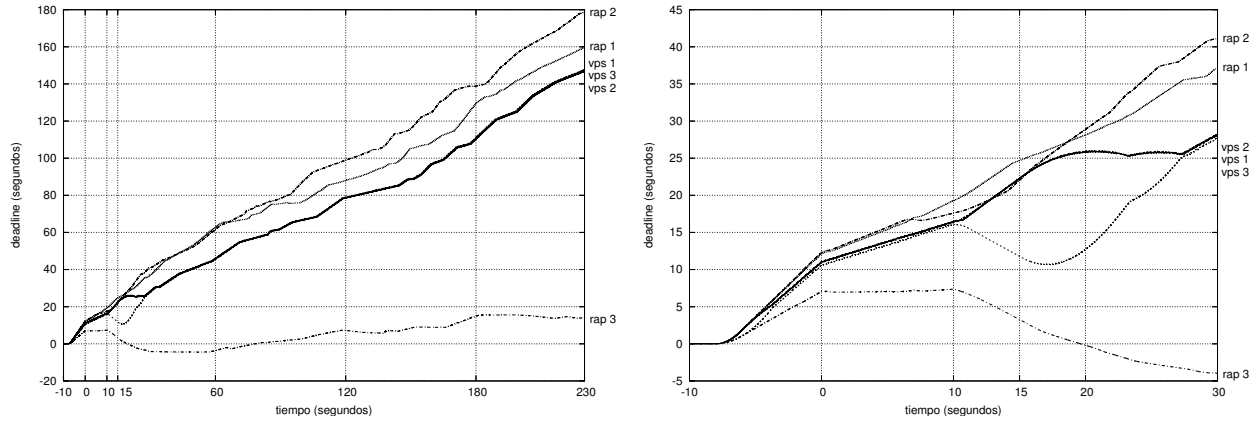
3.1. VPS vs. Aplicación con flujos RAP

A continuación, se muestran las ventajas que logra el VPS presentado, frente a la aplicación con conexiones independientes RAP que no dispone de ningún tipo de planificación de alto nivel.

La figura 5 (a) muestra el resultado de toda la simulación mediante un gráfico "tiempo vs *deadline*". En la figura 5 (b) se observa, con mayor detalle, la reacción del VPS y de RAP, al momento de producirse la congestión en el camino de comunicación con el cliente 3.

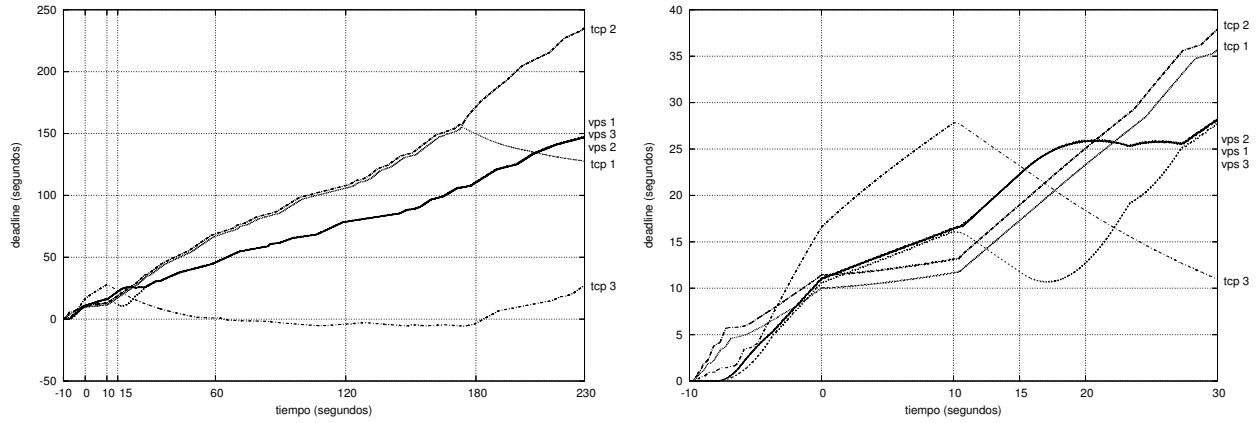
El VPS se recupera rápidamente luego de que el tráfico, que se apropió (entre el segundo 10 y 15) del ancho de banda entre el enlace del nodo n4 al n5, deja de existir. Mientras el flujo del cliente 3 tiene menor *deadline*, el VPS le dedica todos los recursos posibles para mejorar su situación. Como se observa en los gráficos, el VPS disminuye el envío de media a los clientes 1 y 2, logrando un pronunciado ascenso del *deadline* del cliente 3. Aproximadamente en el segundo 25, los tres flujos vuelven a tener el mismo *deadline*. Ninguno de los clientes experimenta interrupciones durante la reproducción de los vídeos y se aprecia un muy buen funcionamiento general del VPS.

Ahora, analicemos el caso de la aplicación que envía el vídeo a los clientes sin ninguna planificación de flujos, donde cada conexión RAP es independiente de las demás. El cliente 3, a los 20 segundos de haber comenzado la reproducción, empieza a experimentar cortes en la visualización del vídeo. En el tiempo 58, el cliente 3 ha tenido cortes en la reproducción por un total de 5 segundos, y recién desde ese momento comienza a recuperarse. Sin embargo, la presencia de los otros dos flujos, impiden que este flujo mejore su transmisión como debería. Note que el cliente 1 tiene menos ancho de banda que el cliente 3 (5 Mbps contra 7 Mbps). No obstante, teniendo el cliente 3 todo el ancho de banda disponible a partir del tiempo 15, no supera en ningún momento la pendiente de crecimiento del *deadline* experimentada por el cliente 1. Esto demuestra, que al nivel de comunicación, se produce una clara injusticia en la distribución del ancho de banda entre las diferentes conexiones.



(a) Simulación completa (b) Momento de congestión en la red

Figura 5: VPS vs. RAP



(a) Simulación completa (b) Momento de congestión en la red

Figura 6: VPS vs. TCP

Si se compara el *deadline* promedio de los tres flujos al finalizar la simulación, el VPS también saca ventaja. El VPS presenta un *deadline* promedio de 144,9 segundos ($(145 + 144, 5 + 145, 1)/3$), mientras que la aplicación con RAP presenta un *deadline* promedio de 117,6 segundos ($(160+178, 7+14, 1)/3$).

3.2. VPS vs. Aplicación con flujos TCP

A continuación, se muestran las ventajas que logra el VPS presentado, frente a la aplicación con conexiones independientes TCP que no dispone de ningún tipo de planificación de alto nivel.

La figura 6 (a) muestra el resultado completo de la simulación mediante un gráfico "tiempo vs *deadline*". En la figura 6 (b) se observa, con mayor detalle, la reacción del VPS y de TCP, al momento de producirse la congestión en el camino de comunicación con el cliente 3.

El comportamiento del VPS en esta simulación ya fue descripto en la comparación efectuada entre el VPS y la aplicación con comunicaciones RAP (ver sección 3.1).

Por lo tanto, queda explicar el comportamiento de la aplicación que envía el vídeo a los clientes sin ninguna planificación de flujos, donde cada conexión TCP es independiente de las demás. El

cliente 3, a los 64 segundos de haber comenzado la reproducción, empieza a experimentar cortes en la visualización del vídeo. En el tiempo 174, el cliente 3 ha sufrido cortes en la reproducción por un total de 6 segundos. Desde este momento, comienza a recuperarse. Es decir, se requirieron 169 segundos para recuperarse de la congestión que duró solo 5 segundos. La presencia de los otros dos flujos, impiden que este flujo mejore su transmisión como se espera. No obstante, el incremento del flujo del cliente 3, es producido junto a una baja del flujo del cliente 1, y un leve aumento del flujo del cliente 2. Al igual que con el protocolo RAP, con TCP se aprecia una injusta distribución del ancho de banda entre las diferentes conexiones.

Si se compara el *deadline* promedio de los tres flujos al finalizar la simulación, el VPS logra mejorar el rendimiento de la aplicación con conexiones TCP. El VPS presenta un *deadline* promedio de 144,9 segundos ($(145 + 144,5 + 145,1)/3$), mientras que la aplicación con TCP presenta un *deadline* promedio de 130,2 segundos ($(128 + 235,7 + 26,9)/3$).

4. CONCLUSIONES Y TRABAJOS FUTUROS

Este artículo es parte de una innovativa línea de investigación enfocada en sistemas de Video bajo Demanda (VoD, Video on Demand) que garanticen una alta calidad de servicio en el dominio de Internet u otras redes de condiciones variables. Con la idea de proveer un servicio de T-VoD (True Video on Demand), con tiempos de servicio bajos, y manteniendo la calidad de la media durante todo el tiempo de servicio, se ha diseñado y desarrollado un servidor de VoD o VPS (Video Proxy Server) que funcione adecuadamente en este entorno.

Un sistema como el de VoD que requiere alta capacidad de transmisión en una red con un modelo de servicio de mejor esfuerzo (best effort), debe incluir un componente que administre las comunicaciones al bajo nivel de la red de tal manera que pueda garantizarse la calidad de servicio (QoS, Quality of Service). Este componente clave dentro del sistema de VoD, denominado Planificador del Tráfico de Red (NTS, Network Traffic Scheduler), proporciona información al Planificador de Canales Lógicos (LCS) sobre el estado de la comunicación con cada uno de los nodos clientes.

De esta manera, el LCS puede planificar (al nivel de aplicación) la entrega de los vídeos a los clientes con conocimiento del estado de la red. La falta de control de flujo al nivel de aplicación, donde a cada flujo se le envía tanta media como se pueda enviar, influye en el funcionamiento de los protocolos de transporte, produciendo una distribución desbalanceada del ancho de banda entre los diferentes flujos. Como se ha demostrado en el presente trabajo, los flujos, al estar en competencia con otros flujos, consumen considerable tiempo para adaptarse a las condiciones de la red.

Se ha implementado un prototipo del VPS, que integra el control de flujo del nivel de aplicación con el control de flujo del nivel de red, y se han realizado experimentaciones, demostrando el correcto funcionamiento y viabilidad para trabajar en una red de condiciones variables como la Internet.

Los trabajos futuros se centran principalmente en consolidar el VPS en una arquitectura distribuida, donde los servidores, distribuidos geográficamente, colaboren para entregar la media a los clientes con la QoS apropiada.

REFERENCIAS

[1] Javier Balladini, Leandro Souza, and Remo Suppi. A network traffic scheduler for a vod server on the internet. *International Conference on Signal Processing and Multimedia Applications (SIGMAP 2006), Setúbal, August 2006*.

[2] Shanwei Cen, Calton Pu, and Jonathan Walpole. Flow and congestion control for internet media streaming applications. Technical Report CSE-97-003, 3, 1997.

[3] Sally Floyd and Kevin Fall. Promoting the use of end-to-end congestion control in the Internet. *IEEE/ACM Transactions on Networking*, 7(4):458–472, 1999.

[4] Sally Floyd, Mark Handley, Jitendra Padhye, and Jorg Widmer. Equation-based congestion control for unicast applications. In *SIGCOMM 2000*, pages 43–56, Stockholm, Sweden, August 2000.

[5] M. Handley, J. Pahdye, S. Floyd, and J. Widmer. TCP Friendly Rate Control (TFRC): Protocol specification, RFC 3448, 2003.

[6] S. Jacobs and A. Eleftheriadis. Real-time dynamic rate shaping and control for internet video applications, Workshop on Multimedia Signal Processing, pages 23–25, June 1997.

[7] Simon S. Lam and Geoffrey G. Xie. Group priority scheduling. *IEEE/ACM Transactions on Networking*, 5(2):205–218, 1997.

[8] Steven McCanne and Sally Floyd. Ns - Network Simulator. http://www.isi.edu/nsnam/ns/, 2005.

[9] Jean M. McManus and Keith W. Ross. Video on demand over atm: Constant-rate transmission and transport. In *INFOCOM*, pages 1357–1362, 1996.

[10] B. Qazzaz, R. Suppi, F. Cores, A. Ripoll, P. Hernandez, and E. Luque. Providing interactive video on demand services in distributed architecture. *Proceedings 29th Euromicro Conference, ISBN: 1089-6503-03*, CL - I:215–222, 2003.

[11] Bahjat Mohammad Khaleel Qazzaz. *Admission Control and Media Delivery Subsystems for Video on Demand Proxy Server*. PhD thesis, Universidad Autónoma de Barcelona, 2004.

[12] R. Rejaie, M. Handley, and D. Estrin. Layered quality adaptation for internet video streaming, IEEE Journal on Selected Areas of Communications (JSAC), Winter 2000.

[13] Reza Rejaie, Mark Handley, and Deborah Estrin. RAP: An end-to-end rate-based congestion control mechanism for realtime streams in the internet. In *Technical report 98-681, CS-USC, august 1998*.

[14] Reza Rejaie, Mark Handley, and Deborah Estrin. RAP: An end-to-end rate-based congestion control mechanism for realtime streams in the internet. In *INFOCOM (3)*, pages 1337–1345, 1999.

[15] Dorgham Sisalem and Henning Schulzrinne. The loss-delay based adjustment algorithm: A TCP-friendly adaptation scheme. In *Proceedings of NOSSDAV*, Cambridge, UK., 1998.

[16] Dorgham Sisalem and Adam Wolisz. LDA+: A TCP-friendly adaptation scheme for multimedia communication. In *IEEE International Conference on Multimedia and Expo (III)*, 2000.

[17] Xin Wang and Ioannis Stavrakakis. Study of multiplexing for group-based quality of service delivery. In *Modelling and Evaluation of ATM Networks*, pages 360–379, 1996.

[18] Zoe Antoniou y Ioannis Stavrakakis. An efficient deadline-credit-based transport scheme for prerecorded semisoft continuous media applications. *IEEE/ACM Transactions on Networking, Vol. 10, No. 5, October 2002*.