Capítulo 3

Mejorando la eficiencia del sistema: Arquitectura Double P-Tree

Resumen

En este capítulo se analiza la eficiencia de la arquitectura Proxy-Tree, utilizando un modelo analítico basado en la distancia media de servicio que nos permite comparar la efectividad de las arquitecturas distribuidas. A partir del análisis de los resultados obtenidos se confirma la importancia de la conectividad en las arquitecturas distribuidas y se propone una variante de la arquitectura original, denominada Double P-Tree. En este capítulo, se adapta el modelo analítico de la arquitectura Proxy-Tree teniendo en cuenta las características incorporadas en la arquitectura propuesta y se evalúa su escalabilidad y su eficiencia.

3.1 Introducción

La evaluación de la arquitectura P-Tree realizada en el capítulo anterior ha puesto de manifiesto unas cualidades óptimas para el diseño de sistemas de VoD de gran escala: escalabilidad, tolerancia a fallos y balanceo de carga, entre otras. Sin embargo, para que la arquitectura distribuida propuesta se pueda convertir en una seria alternativa para la construcción de sistemas LVoD es necesario mejorar algunos aspectos de la misma.

En concreto, el principal punto débil de la arquitectura P-Tree lo podemos localizar en la reducción de prestaciones (ancho de banda efectivo) del sistema respecto a otras arquitecturas alternativas. Ésta reducción en la eficiencia del sistema, es el sacrificio que hay que pagar para obtener un sistema distribuido escalable y tolerante a fallos. Sin embargo, no nos resignamos a no intentar mejorar la eficiencia del sistema P-Tree.

En este capítulo vamos a analizar las causas de este problema e intentaremos subsanarlo mediante la propuesta de una topología más eficiente.

3.2 Análisis de la eficiencia de la arquitectura Proxy-Tree

Para mejorar el rendimiento de la arquitectura, primero es necesario analizar las causas de su menor eficiencia respecto a otras arquitecturas similares.

El principal factor que penaliza el rendimiento de las arquitecturas distribuidas como la nuestra, es el ancho de banda de red adicional requerido, debido a las peticiones que no se pueden servir localmente. Mientras las peticiones que se sirven localmente consumen recursos de la red local, las peticiones que deben ser atendidas desde servidores-proxy adyacentes, requieren ancho de banda en cada una de las redes que se tienen que cruzar.

Una buena aproximación para medir la penalización de ancho de banda de red requerida por los sistemas distribuidos, es la *distancia media de servicio*. Esta distancia se puede definir como el número de redes (sin tener en cuenta las redes de conexión de los usuarios) que deben cruzar en promedio los flujos de información utilizados para servir las peticiones en el sistema.

Recordemos que las peticiones que se pueden servir localmente se dice que están a distancia-1 ya que el stream de servicio únicamente requiere ancho de banda en la red local. Una petición atendida desde un servidor vecino a la red local del usuario, se ubica a distancia 2. Los streams servidos a distancia-2 necesitan cruzar primero la red del servidor adyacente y a continuación la red local a la cual esta conectado el usuario. En el caso de utilizar técnicas

unicast de transmisión sin comparticion (con las que hemos estado trabajando en esta primera parte de la investigación), una petición servida a distancia-2, requiere el doble de ancho de banda de red para ser servida que una petición a distancia-1.

La distancia media de servicio es muy importante, ya que permite evaluar a priori el ancho de banda efectivo de red que soporta un sistema de VoD distribuido. Esta métrica, también nos permite comparar entre si, distintos sistemas distribuidos, evaluando su capacidad de servicio respecto a otras arquitecturas de referencia.

En un sistema centralizado ó un sistema basado en servidores independientes, la distancia media de servicio siempre es 1; todas las peticiones se pueden atender localmente ya que el servidor tiene una copia completa del catálogo de contenidos. En este tipo de arquitecturas el acho de banda de red efectivo es equivalente al ancho de banda físico de red.

Sin embargo, en un sistema distribuido de VoD la distancia media de servicio es mayor debido a que no todas las peticiones se pueden servir localmente. Un porcentaje de las peticiones se tendrán que servir a distancias mayores, consumiendo más recursos de red. Esto conlleva, en un primer análisis (y sin tener en cuenta otros aspectos favorables a la arquitectura), una reducción del ancho de banda de red efectivo. Por ejemplo, un sistema de VoD distribuido (DVoD) con una distancia media de servicio de 1.5, implica que en promedio, cada stream requiere un 50% más de ancho de bada de red para ser transmitido desde el servidor hasta el cliente.

3.2.1 Cálculo de la distancia media de servicio

La distancia media de servicio se puede calcular realizando la media de las distancias de servicio para cada uno de las redes locales que componen el sistema distribuido de VoD.

La distancia de servicio de una red local se puede calcular estáticamente (sin tener el sistema en funcionamiento, ni teniendo en cuenta restricciones de ancho de banda), como la distancia media necesaria para poder acceder a cada uno de los videos que conforman el catálogo del sistema, desde cada una de las redes del sistema. Los contenidos que se pueden atender desde el servidor local están a una distancia 1, los contenidos accedidos desde servidores adyacentes están una distancia 2 y así sucesivamente.

En el caso de que al calcular la distancia media de servicio se tenga en cuenta la popularidad de los contenidos, entonces estaremos hablando de una *distancia media de servicio ponderada*. Esta última métrica, es la que mejor se ajusta al rendimiento real de un sistema de VoD, ya que también tiene en cuenta el comportamiento de los usuarios.

Para evaluar la distancia media de servicio de cualquiera sistema DVoD, asumiremos que todos los contenidos multimedia del sistema tienen el mismo tamaño y que la capacidad de

almacenamiento del servidor es siempre un múltiplo de este tamaño. Cada servidor del sistema tiene suficiente capacidad para almacenar V_S videos de los N disponibles en el catálogo.

En la arquitecturas propuesta, la cantidad de almacenamiento dedicada a cada uno de los esquemas de gestión (caching y mirroring) puede llegar a tener una importante influencia sobre la distancia de servicio. Por lo tanto, ambos parámetros se tendrán en cuenta a la hora de realizar el análisis. De esta forma, asumiremos que el almacenamiento del servidor se distribuye entre V_c videos gestionados mediante caching y V_m videos asignados al mirror. En el caso que un determinado servidor no utilice uno de los esquemas, bastará con asignarle un valor nulo al parámetro correspondiente.

Nuestro propósito es poder evaluar las distancias de servicio medias para cualquier arquitectura distribuida de VoD, de forma que podamos comparar la eficiencia de diversas arquitecturas. Para alcanzar este objetivo, necesitamos modelar de alguna forma los diferentes tipos de arquitecturas distribuidas, utilizando para ello la conectividad media de la topología utilizada (*C*), y la distancia máxima de servicio dentro del sistema (*MD*).

La distancia media de servicio se puede calcular en función del porcentaje de peticiones gestionadas en cada una de las distancias (R_i), de la siguiente forma:

$$D_{ms} = \sum_{i=1}^{MD} R_i \qquad (3.\mathbf{1})$$

El porcentaje de peticiones gestionadas a distancia *i*, identifica el volumen de peticiones que al menos tienen que cruzar *i* redes locales para poder ser servidas. R_i tiene que estar acotada entre 1 y 0 (100% ó 0% de las peticiones). Como máxima distancia de servicio, MD, se puede utilizar la longitud del camino máximo entre las dos redes locales más alejadas de la topología.

Por definición, todas las peticiones tendrán que cruzar como mínimo una red para ser atendidas (la red local). Por lo tanto, el porcentaje de peticiones que al menos tiene que utilizar una red (distancia 1), R_1, es del 100%,

$$R_1 = R_{Total} = 1 \qquad (3.\mathbf{2})$$

Las peticiones que al menos deben cruzar 2 redes (R_2) son todas aquellas que no han podido ser atendidas localmente mediante su servidor local. El volumen de estas peticiones se pueden calcular mediante la probabilidad de que una petición no sea atendida por el servidor local. Esta probabilidad se puede desglosar entre la probabilidad de ser servida desde la cache ($R_{ATcache}$) y la probabilidad de hacerlo desde el mirror local ó mirror distribuido a distancia 1 ($R^1_{ATmirrorD}$):

$$R_2 = R_{Total} - R^1_{AT} = 1 - (R_{ATcache} + R^1_{ATmirrorD}) \qquad (3.\mathbf{3})$$

Sustituyendo el mirror local por el mirror distribuido a distancia d-1, podemos generalizar la formula anterior para cualquier distancia. De esta forma la probabilidad de que una petición requiera cruzar *d* redes, se puede evaluar como:

$$R_d = R_{Total} - R_{AT}^{d-1} = 1 - (R_{ATcache} + R_{ATmirrorD}^{d-1}) \quad (3.4)$$

En la ecuación anterior hemos asumido que todas las caches de los servidores tendrán los mismos contenidos (los más populares). Por lo tanto la probabilidad de que una petición, que previamente no ha podido ser atendida por la cache de su servidor local, pueda ser servida desde una cache remota es prácticamente nula. Teniendo en cuenta este criterio, las peticiones gestionadas remotamente solo pueden servirse desde los mirrors de los servidores remotos (mirror distribuido).

La probabilidad de acierto en la cache se puede calcular como la suma de la probabilidad de acceso de los V_c videos más populares. Asumiendo que la frecuencia de acceso a los videos sigue una distribución Zipf, entonces $R_{ATcache}$ se puede evaluar de forma equivalente a la ecuación 2.3 del capitulo anterior (apartado 2.3):

$$R_{ATcache} = P_C = \sum_{m=1}^{Vc} \frac{1}{m^z \cdot \sum_{i=1}^{N} 1/i^z} \quad (3.5)$$

La probabilidad de que una petición pueda ser atendida desde el mirror distribuido a distancia *d* depende básicamente de su tamaño. La capacidad de un mirror distribuido a distancia-d se puede evaluar en función del número de servidores que lo componen, como $\sum_{i=0}^{d-1} C^i$, multiplicado por la capacidad individual del mirror de cada uno de los servidores-proxy (V_m).

Una vez calculado el número de contenidos que componen el mirror distribuido, únicamente tenemos que multiplicarlos por la probabilidad media de acceso a los contenidos del mirror (P_{am}),

$$R_{ATmirrorD}^{d} = V_M * P_{am} * \sum_{i=0}^{d-1} C^i = V_M * P_{am} * \frac{1-C^d}{1-C} \quad (3.6)$$

La probabilidad media de acceso de los contenidos de un mirror (P_{am}) se puede evaluar utilizando la expresión (2.8) del capitulo anterior.

Utilizando las expresiones (3.2), (3.4) y (3.6), podemos calcular el porcentaje de peticiones gestionadas a distancia *d* ó superior del siguiente modo:

$$R_d = \left\{ \begin{array}{ll} 1-(P_{ac}+V_M * P_{am} * \frac{1-C^d}{1-C}), & si\ (P_{ac}+V_M * P_{am} * \frac{1-C^d}{1-C}) < 1 \\ 0, & si\ (P_{ac}+V_M * P_{am} * \frac{1-C^d}{1-C}) > 1 \end{array} \right\} \forall d > 1 \qquad (3.7)$$

Esta expresión calcula el porcentaje de peticiones que necesitan al menos de *d* redes para ser gestionadas. En el caso de que todas las peticiones puedan ser gestionadas a distancia d-1, se cumple que:

$$(P_{ac}+V_M * P_{am} * \frac{1-C^d}{1-C}) > 1$$

y la expresión anterior devuelve un 0.

Finalmente, si tenemos en cuenta las peticiones a distancia 1 y utilizando la expresión (3.1), podemos aproximar la **distancia media de servicio** (D_{ms}) de un sistema distribuido de VoD como:

$$D_{ms} = 1 + \sum_{i=2}^{MD} R_i \qquad (3.8)$$

3.2.2 Reducción de la distancia de servicio en las arquitecturas distribuidas

Para analizar la distancia de servicio de la arquitectura P-Tree, vamos a calcular la distancias medias de servicio (D_{ms}), mediante la expresión (3.8), para las principales arquitecturas LVoD.

El cálculo para las arquitecturas centralizadas y de servidores independientes es inmediato (la distancia media de servicio es 1). Sin embargo, para las arquitecturas distribuidas (servidores-proxy de un nivel y Proxy-Tree) vamos a realizar los cálculos paso a paso de forma que sea más sencillo comprender el análisis posterior de los resultados.

Primero vamos a desglosar el cálculo para el calculo realizado para la arquitectura de servidores-proxy de un nivel, en la cual la distancia máxima de servicio (*MD*) es 2 y la conectividad media 1 (*C*).

$$\begin{aligned} D_{ms} &= 1 + \sum_{i=2}^{2} R_i \\ &= 1 + R_2 \\ &= 1 + \left[1 - (R_{ATcache} + R^1_{ATmirrorD})\right] \end{aligned}$$

La probabilidad de acierto en el mirror es 0 ya en esta arquitectura todo el almacenamiento de los servidores-proxy se gestiona mediante el esquema de cache. La probabilidad de acierto en la cache se puede obtener mediante la suma de los 30 videos más populares, en este caso es del 63.5% y por lo tanto tendremos:

$$D_{ms} = 1 + [1 - (0.635 + 0)] = 1.365$$

En el caso de la arquitectura P-Tree tenemos una conectividad media de la topología (C) de 1,968 y suponemos una distribución del almacenamiento de 12 videos asignados a la cache (con una probabilidad de acierto total del 34,37%) y 18 videos asignados al mirror y una distancia máxima de servicio de 11 (6 niveles * 2 –1):

$$\begin{aligned} D_{ms} &= 1 + \sum_{i=2}^{11} R_i \\ &= 1 + R_2 + R_3 + R_4 + \cdots + R_{11} \end{aligned}$$

El porcentaje de peticiones que se atienden a distancia 2 ó superior se puede calcular utilizando la expresión (3.3) y (3.6):

$$R_2 = 1 - (R_{ATcache} + R^1_{ATmirrorD})$$

$$R_2 = 1 - (0.343 + \left[V_M * P_{am} * \frac{1 - C^1}{1 - C} \right])$$

$$R_2 = 1 - (0.343 + [18 * 0.01 * 1]) = 0.48$$

El resultado anterior significa que el 52% de todas las peticiones se pueden gestionar localmente y que el resto (48%) al menos requiere una distancia de servicio de 2. De forma equivalente podemos calcular las peticiones que se atienden a distancia 3:

$$R_3 = 1 - (R_{ATcache} + R^2_{ATmirrorD})$$

$$R_3 = 1 - (0.343 + \left[18 * 0.01 * \frac{1 - 1.968^2}{1 - 1.968} \right])$$

$$R_3 = 1 - (0.343 + [18 * 0.01 * 2.968]) = 0.122$$

El cálculo de las peticiones que se deben atender a distancia 4 sale negativo debido a que todas las peticiones se puede gestionar a distancia 3 ó inferiores. Por lo tanto, la distancia media de servicio final para la arquitectura P-Tree será:

$$D_{ms} = 1 + 0.48 + 0.122 = 1.602$$

En la tabla 3-1 resumimos las distancias medias de servicios para las 4 arquitecturas analizadas. De estos resultados, podemos sacar la conclusión de que a medida que incrementamos el grado de distribución de la arquitectura, también se incrementa la distancia de servicio de las peticiones.

Tabla 3-1. Distancia media de servicio para las principales arquitecturas LVoD

Centralizado	Servidores independientes	Un nivel de servidores-proxy	Proxy- Tree
1	1	1,365	1,602

Analizando las distancias de servicio teóricas de las arquitecturas de servidores-proxy de un nivel y P-Tree, podemos darnos cuenta de la causa que provoca la reducción de prestaciones de nuestra arquitectura: la distancia media de servicio. La arquitectura P-Tree tiene una distancia de servicio que es un 18% mayor que la distancia de servicio de los servidores-proxy de un nivel y un 60% más grande que la óptima (sistemas centralizados). Ésta desviación implica que en promedio cada petición consumirá entre un 18% y un 60% más recursos de red.

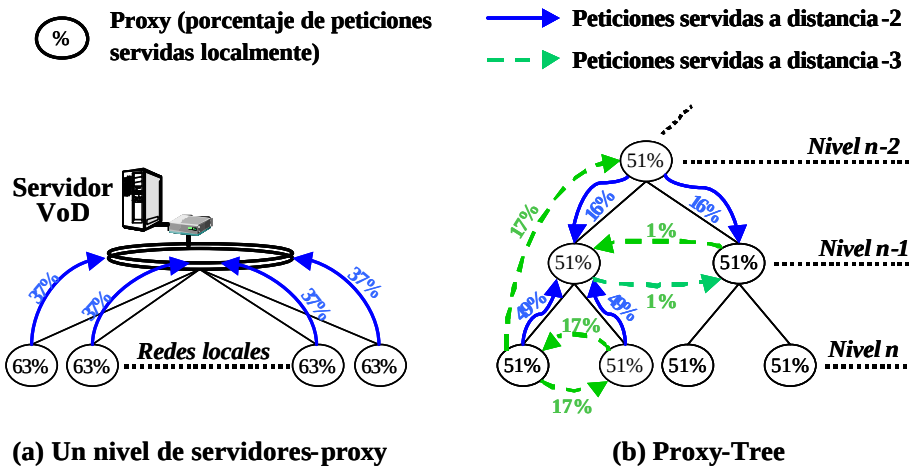
Si queremos mejorar el rendimiento estructural (sin incrementar los recursos) de nuestra arquitectura, debemos reducir la distancia media de servicio de las peticiones de nuestro sistema.

El principal factor que afecta a la distancia de servicio de los sistemas distribuidos de VoD son los fallos en los servidores-proxy y su distribución por la topología. En los sistemas de un nivel de servidores-proxy la probabilidad de fallo en los servidores-proxy depende de su capacidad de almacenamiento y del patrón de acceso de los usuarios. En la arquitectura P-Tree, el porcentaje de peticiones que no se pueden servir localmente (fallo en el servidor-proxy) además se ve afectado por la distribución del almacenamiento entre los dos esquemas de gestión y por la capacidad de los mirror distribuidos a los cuales puede acceder un proxy.

Para facilitar el análisis de la influencia de estos parámetros sobre la distancia de servicio de las arquitecturas de un nivel de servidores-proxy y P-Tree, hemos dibujado en la figura 3-1 como se distribuye la gestión de las peticiones entre los distintos servidores-proxy. Usando la ecuación (3.8) la figura muestra el porcentaje de peticiones que se atienden localmente y como se distribuye el resto de peticionen entre los servidores-proxy adyacentes.

Comparando ambos sistemas podemos distinguir los dos principales factores que afectan a la distancia de servicio de la arquitectura P-Tree: el porcentaje de peticiones que se tienen que servir localmente y las peticiones servidas a distancia-3.

Como se observa en la figura, los servidores-proxy de la arquitectura de un nivel pueden atender el 63% de las peticiones localmente (enviando el resto al servidor principal) en contraposición del 51% en la arquitectura P-Tree (debido a la distribución del almacenamiento



(a) Un nivel de servidores-proxy **(b) Proxy-Tree**

Figura 3-1. Distribución de las peticiones remotas en las arquitecturas basadas en servidores-proxy

entre dos esquemas de gestión de contenidos). Por lo tanto, la segmentación del almacenamiento en P-Tree implica una mayor probabilidad de fallos en los servidores-proxy, un incremento del trafico en las redes locales y una reducción del rendimiento.

Si ahora nos fijamos en la distribución de las peticiones remotas, podemos constatar que en el sistema de servidores-proxy de un nivel, gracias al servidor principal, todos los fallos en los servidores-proxy pueden ser atendidos como máximo a distancia 2.

Sin embargo, el sistema P-Tree analizado no permite que todas las peticiones remotas puedas ser atendidas desde el mirror distribuido a distancia-2 y algunas de ellas se deben servir a distancia 3. El volumen de estas peticiones depende del nivel en el que se localiza el proxy. En los niveles intermedios únicamente el 1% de todas las peticiones se tiene que servir desde tan lejos, mientras que en el último nivel este porcentaje alcanza el 16% de las peticiones.

Este desequilibrio entre la eficiencia de los servidores-proxy del último nivel y los niveles intermedios son debidas a la capacidad del mirror distribuido a distancia 2. Los servidores-proxy ubicados en el último nivel carecen de hijos, por lo tanto tienen que enviar todas los fallos hacia los servidores-proxy de niveles superiores. El mirror distribuido de estos servidores-proxy consta de un único proxy a distancia-2 (en contraposición de los 3 que disponen los servidores-proxy situados en niveles intermedios), de 2 servidores-proxy a distancia-3 (6 para los servidores-proxy situados en niveles intermedios).

La pobre conectividad del último nivel incrementa el ancho de banda requerido para atender los fallos en los servidores-proxy, ya que el trafico asociado con estas peticiones es mayor que en otros niveles (16% frente al 1%). Además, esta penalización se acentúa más debido a que el último nivel es el más grande de una topología en árbol, llegando a tener casi tantos nodos como

el resto de niveles juntos. Este efecto provoca también el efecto de desbalanceo de los anchos de banda requeridos para las redes locales de los niveles penúltimo y antepenúltimo, como ya mencionados en el apartado 2.5.4 del capítulo anterior.

La combinación de un menor número de peticiones gestionadas localmente, más el elevado número de peticiones que se deben atender desde servidores demasiado alejados provoca un incremento de la distancia de servicio de la arquitectura P-Tree respecto al sistema de un nivel de servidores-proxy.

Para tratar de reducir la distancia de servicio de la arquitectura P-Tree, vamos a estudiar las expresiones (3.7) y (3.8) que modelan esta distancia en las arquitecturas distribuidas. Un análisis de estas expresiones nos permite identificar dos diferentes alternativas para reducir la distancia de servicio:

1. *Incrementar la probabilidad de acierto de la cache (p_{am})*

   Para lograr que la cache sea más efectiva, la única formula posible es incrementar su tamaño. Esta solución no es valida, porque el aumento de la eficiencia del sistema lo queremos obtener sin aumentar la capacidad de almacenamiento de los servidores-proxy.

   Tampoco se puede transferir capacidad de almacenamiento desde el mirror hacia la cache, ya que las distribuciones escogidas son las que permiten obtener el mejor rendimiento. Asignar almacenamiento del mirror a la cache implicaría reducir la eficiencia del mirror distribuido y a la larga, incrementar la distancia de servicio.

2. *Aumentar la eficiencia del mirror distribuido*

   La primera posibilidad para aumentar la eficiencia del mirror distribuido pasa por aumentar el tamaño de los mirror parciales de los servidores-proxy (parámetro V_M). Esta solución no es factible por el mismo razonamiento que para la cache.

   La otra alternativa que nos queda para incrementar la eficiencia del mirror distribuido pasa por incrementar el tamaño del mirror distribuido. Este tamaño depende de dos factores: de la capacidad dedicado al mirroring en cada servidor y del número de servidores / servidores-proxy que forman parte del mirror distribuido. Según podemos ver en la expresión (3.7), el número de servidores-proxy que componen un mirror distribuido esta en función de la conectividad media de la topología. Por lo tanto, el incremento de la capacidad del mirror distribuido se puede alcanzar incrementando la conectividad de la topología (parámetro *C*).

   El aumento de la conectividad de la topología no requiere necesariamente incrementar los recursos requeridos por el sistema y, por lo tanto, es compatible con nuestras premisas iniciales.

3.2.3 Aumento de la conectividad de la topología P-Tree

En este apartado vamos a proponer dos técnicas para incrementar la conectividad de la topología, pero sin tener que sacrificar las principales cualidades de la arquitectura P-Tree. La primera técnica se basa en incrementar el orden del árbol y la segunda trata de incrementar la interconexión entre si de las redes locales de la arquitectura.

Incremento del orden del árbol

La primera alternativa consiste en la utilización de árboles terciaros ó cuaternarios para la topología en vez del árbol binario utilizado hasta el momento.

Para evaluar la bondad de esta aproximación vamos a calcular la conectividad media de la nueva topología obtenida y la vamos a comparar con la conectividad media de la topología P-Tree original.

La conectividad media de una topología se puede calcular dividiendo la suma de la conectividad individual (C_i) de cada uno de los nodos de la topología ($\sum_{i=1}^{N} C_i$) entre el número de nodos del sistema (N):

$$C_M = \frac{\sum_{i=1}^{N} C_i}{N} \qquad (3.9)$$

En la figura 3-2, mostramos la conectividad media obtenida para un árbol terciario (fig. 3-2b) y la comparamos con la conectividad original de la arquitectura P-Tree (fig. 3-2a) utilizando una arquitectura con 31 nodos.

Podemos comprobar que la conectividad media no mejora, debido a que la topología resultante tiene un último nivel más grande. Los nodos ubicados en el último nivel son los que

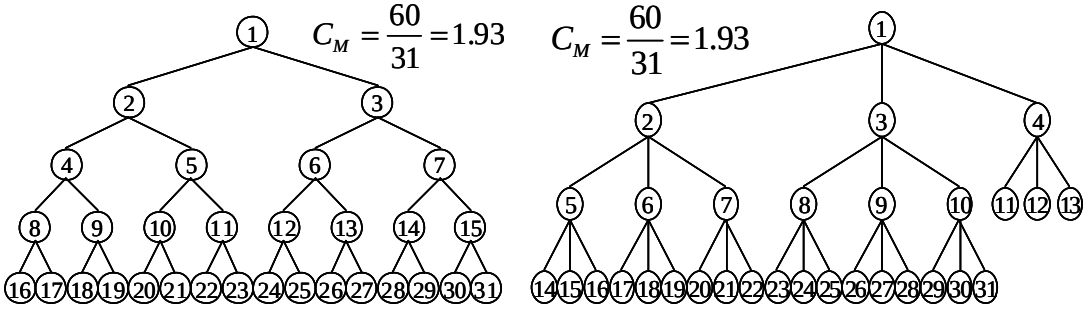


Figura 3-2. Conectividad media de topologías en árbol

peor conectividad tienen en una topología en árbol (ya que no tienen hijos), penalizando considerablemente la mejor conectividad del resto de niveles.

Este problema con los nodos del último nivel se agrava considerablemente si tenemos en cuenta que el número de nodos sin hijos en un árbol binario es casi tan grande como el resto de nodos de la topología. En un árbol terciario, la relación existente entre los nodos del último nivel y el resto de niveles es más del doble. Esta pobre conectividad se ve corroborada por los resultados obtenidos por la arquitectura P-Tree de orden 3 y 4, mostrados en el apartado 2.5.1 del capítulo anterior. El ancho de banda efectivo del sistema para estas configuraciones no mejora la eficiencia obtenida con la topología binaria.

Redes locales hermanas

La segunda alternativa para aumentar la conectividad consiste en la interconexión de diversas redes locales de un mismo nivel de la topología. Las redes locales interconectadas de esta forma se denominan redes hermanas y los servidores conectados a estas redes locales, servidores hermanos. Este esquema nos ofrece la posibilidad de incrementar la conectividad de la topología de forma flexible y en mayor medida que la obtenida mediante las otras alternativas.

En la figura 3-3 mostramos la conectividad de la topología mediante los servidores hermanos para un sistema compuesto por 31 nodos y 1 hermano. Como podemos ver esta topología consigue incrementar considerablemente la conectividad media, pasando de 1.93 (en la figura 3.2a) a 2.9 y consecuentemente, se reduce la distancia de servicio del sistema.

La nueva topología permite reducir la distancia de servicio de las peticiones de los usuarios. La mejora más evidente se produce en la reducción de las distancias de las peticiones que se tienen que atender a distancia 3 y superiores. Con el aumento de la conectividad, también

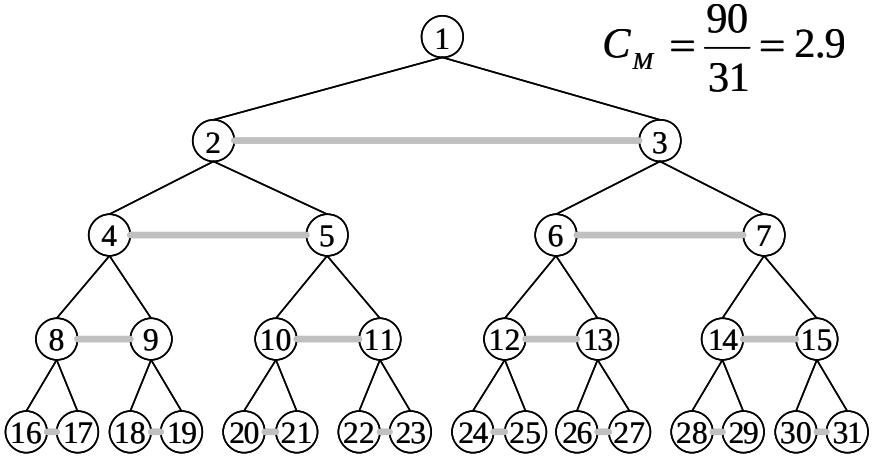


Figura 3-3. Conectividad media de topología árbol con un servidor hermano

conseguimos incrementar el número de servidores vecinos, que son los que componen principalmente el mirror distribuido a distancia 2. Al tener más capacidad, este mirror distribuido es más eficiente y es más probable que sea capaz de dar cabida a todos los contenidos del catálogo. Gracias a la mejora de la conectividad se consigue reducir considerablemente el volumen de peticiones que se tienen que gestionar a distancia 3 ó superiores.

La mejora de la conectividad no solo repercute en el rendimiento del mirroring, sino que también puede llegar a incrementar la capacidad de la cache y con ello el número de peticiones que se pueden atender localmente. En el capitulo anterior (apartado 2.5.2), vimos como a la hora de distribuir el almacenamiento del proxy, se sobredimensiona el tamaño del mirror hasta conseguir reducir al mínimo el posible el trafico a distancia-3. Mediante el incremento de la conectividad de la topología ya podemos alcanzar este objetivo sin necesidad de sobredimensionar la capacidad asignada al esquema de mirroring. Este hecho nos permite desviar la capacidad de almacenamiento sobrante del mirror hacia el esquema de caching, incrementando su eficiencia.

La topología resultante de la interconexión de las redes locales permite un incremento flexible y gradual de la conectividad en función del número de hermanos. Esta característica nos permitirá adaptar la conectividad de la topología a las necesidades del sistema de VoD.

Debido a estas ventajas, se ha escogido esta aproximación para aumentar la conectividad de la topología y con ello el rendimiento del sistema. La nueva arquitectura resultante, denominada Double P-Tree, se presenta en el siguiente apartado.

3.2 Arquitectura Double P-Tree

Utilizando el concepto de los servidores hermanos, hemos propuesto la arquitectura mostrada en la figura 3-4. Esta nueva arquitectura se denomina Double P-Tree debido a que los servidores hermanos forman un segundo árbol dentro de la topología P-Tree original [Cor02] [Cor02b].

La nueva topología introduce un nuevo parámetro en la configuración del sistema: el número de hermanos. El sistema mostrado en la figura 3-4, muestra una arquitectura Double P-Tree con 3 hermanos, lo cual significa que siempre que sea posible (salvo en los niveles 0 y 1) cada red local estará interconectada con otras 3 redes locales del mismo nivel.

Como se puede ver en la figura, esto no es posible en los 2 primeros niveles, que no disponen de suficientes nodos como para completar la conexión de las redes locales hermanas.

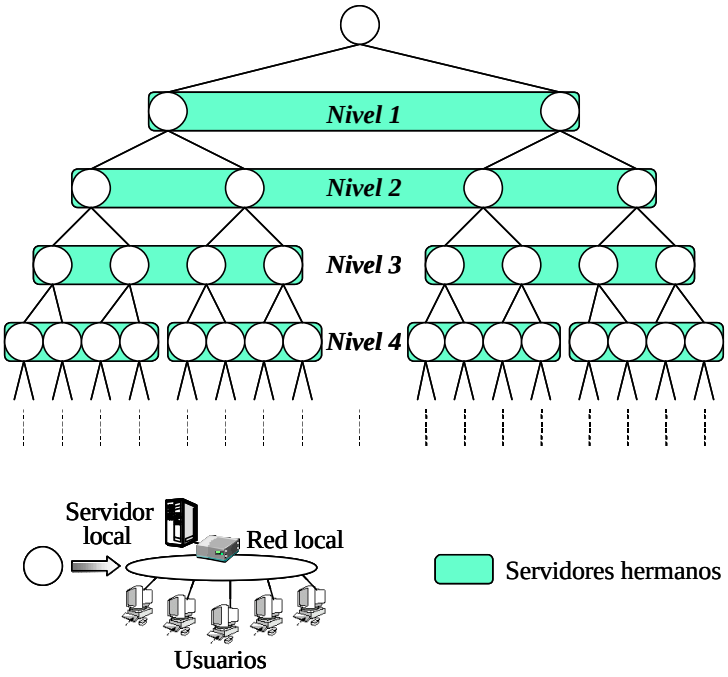


Figura 3-4. Arquitectura Double P-Tree

Selección de los servidores hermanos

La arquitectura Double P-Tree admite la selección de cualquier número de servidores hermanos, aunque normalmente se prefiere utilizar un número impar de hermanos de forma que se obtenga una topología mas homogénea.

Sin embargo, en la práctica el número de hermanos de la topología esta acotado por el número de puertos disponibles en el switch de la red local. Cada nuevo hermano que se incluye a la topología implica reducir el número de puertos disponibles para la conexión de los usuarios.

Se necesita llegar a un compromiso entre los puertos asignados a la topología y los puertos asignados a los usuarios. Los puertos de los usuarios son los que en promedio menos tráfico tienen que soportar. A través de la experimentación realizada se observa que asignando menos de la mitad de los puertos del switch para la conexión de los usuarios ya es suficiente para evitar que el sistema se sature.

En la tabla 3-2, podemos ver como a medida que se incrementa el número de hermanos, también se aumenta la conectividad de la topología y con ello la eficiencia teórica de la arquitectura Double P-Tree.

Tabla 3-2. Conectividad media de las topologías Double P-Tree

Double-PTree con 1 hermano	Double-PTree con 3 hermanos	Double-PTree con 5 hermanos	Double-PTree con 7 hermanos
2.9	4.7	5.8	7.8

La otra cuestión que se nos plantea con respecto a la nueva topología, es el criterio que se debe seguir para agrupar las redes locales entre si, y de esta forma definir los servidores hermanos de un determinado nodo. Existen dos posibles criterios para seleccionar los servidores hermanos: seleccionar redes locales adyacentes ó seleccionar redes locales equidistantes entre si.

Mediante la selección de las redes locales más cercanas se obtiene la topología ya mostrada en la figura 3-4. La principal ventaja de esta selección estriba en que permite la creación de una topología más compacta que se puede distribuir físicamente de forma más fácil. La arquitectura resultante permite fraccionar la topología en diversos sub-árboles, los cuales se pueden ubicar de forma separada y que únicamente requieren de 4 enlaces para mantener la conexión entre si.

El segundo criterio selecciona las redes locales de forma equidistantes dentro del nivel, logrando la topología que se muestra en la figura 3-5. Esta implementación de los servidores hermanos tiene diversas ventajas. La primera consiste en el incremento de la capacidad del

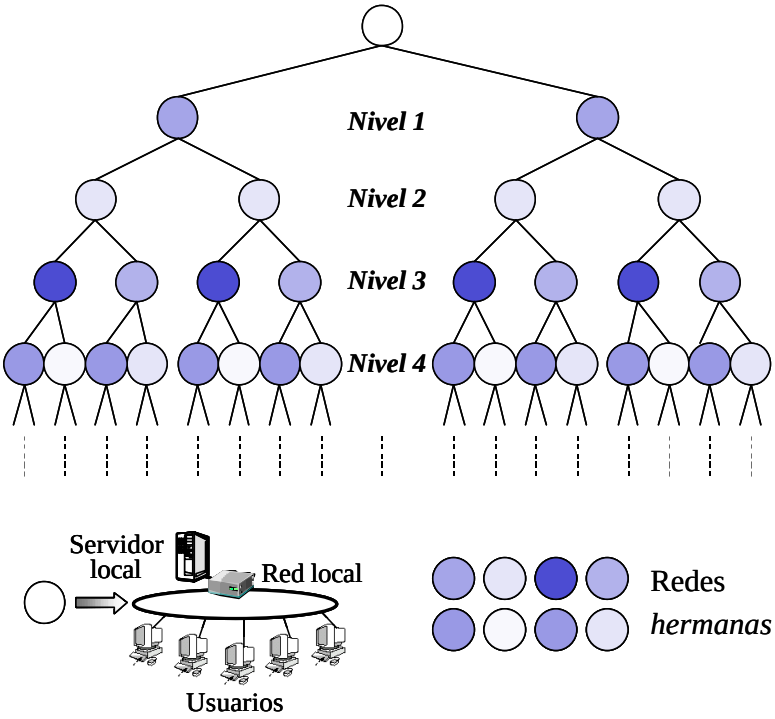


Figura 3-5. Selección redes hermanas equidistantes entre si

mirror distribuido a distancia-3. Al estar los servidores hermanos más separadas no hay tanta replicación entre los servidores situados a distancia 3 y por lo tanto la capacidad del mirror distribuido es mayor que en la configuración anterior. Sin embargo, esta característica al final no resulta tan interesante, debido a que una de los principales objetivos alcanzados con la arquitectura Double P-Tree es minimizar el número de peticiones que requieren servirse a distancia 3. Por esta razón, el incremento del mirror distribuido a distancia 3 obtenido mediante este criterio de selección de hermanos no supone una mejora decisiva.

Otra característica interesante de esta segunda configuración es la mejor tolerancia a fallos del sistema. Con la configuración anterior se debe producir el fallo simultaneo de 4 enlaces para dejar un subárbol de la topología totalmente aislado. Sin embargo con la selección de los servidores hermanos de forma equidistante, el número de enlaces que se tienen que romper para que un sub-árbol quede totalmente aislado se incrementa de forma considerable.

Sin embargo, la misma característica que permite incrementar la tolerancia a fallos también obliga a que el número de enlaces requeridos entre las posibles distribuciones geográficas de cada uno de los sub-árboles se incremente considerablemente. Por esta razón, nosotros hemos preferido una configuración más sencilla de implementar como la selección de los servidores hermanos adyacentes.

Gestión de la arquitectura Double P-Tree.

En esta nueva arquitectura, desaparece definitivamente el concepto de servidor central ó red principal. La arquitectura Double P-Tree elimina el servidor central, distribuyendo su funcionalidad (como repositorio de los videos del catálogo del sistema y gestor del sistema de VoD) entre todos los servidores locales. De esta forma, obtenemos un sistema totalmente homogéneo, en el cual cada nodo de la arquitectura esta compuesto por una red local a la cual se conectan un conjunto de usuarios y un servidor local.

La arquitectura mantiene la gestión combinada de los contenidos en los servidores locales mediante las políticas de caching y mirroring.

3.2.1 Implementación de la topología Double P-Tree

Es importante resaltar que la implementación de la topología Double P-Tree no requiere la inclusión de nuevas redes. La nueva topología se alcanza interconectando entre si los switches de las redes locales, tal y como se muestra en la figura 3-6.

En la figura se muestran los puertos reservados en los switches para implementar la topología y como se interconectan entre ellos. Igual que en la arquitectura P-Tree, en la arquitectura

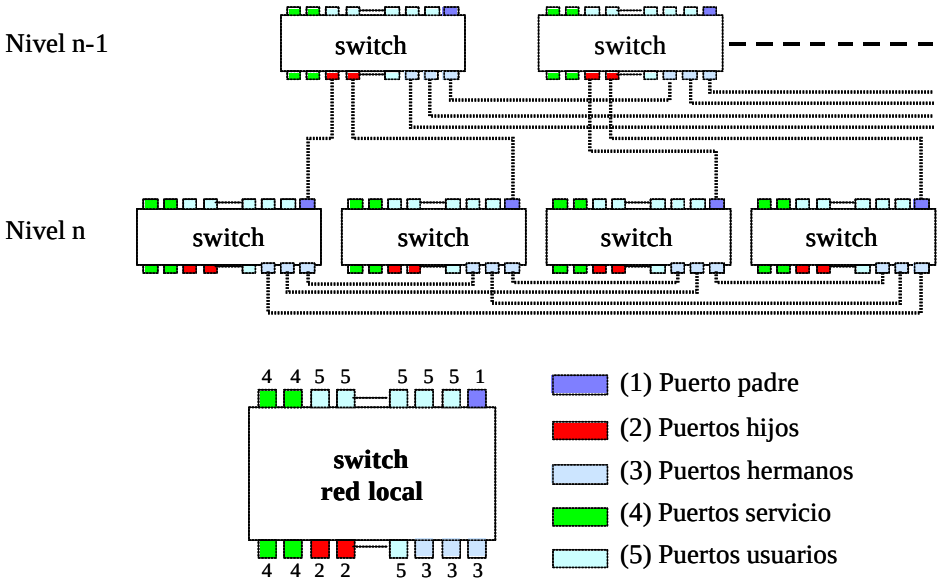


Figura 3-6. Implementación topología Double P-Tree

Double P-Tree se reserva la mayoría de los puertos para conectar las redes de los usuarios. Cada red local dispone de un puerto (puerto padre) para conectarse con el nivel superior de la topología, varios puertos (puertos hijos) dependiendo del orden del árbol para conectarse con el siguiente nivel de la topología.

Para implementar los servidores hermanos se reservan varios puertos (puertos hermanos), dependiendo del número de hermanos de la arquitectura. Cada uno de estos puertos se utiliza para enlazar con cada una de las redes locales hermanas.

Por último, la arquitectura admite la posibilidad de conectar los servidores locales a la red utilizando diversos puertos (puertos de servicio). Esta técnica permite aumentar el ancho de banda de red disponible para el servidor y en algunos casos, reducir los requisitos de ancho de banda para las redes locales (como se explicará en el siguiente apartado).

En la figura, vemos el esquema de interconexión para una arquitectura Double P-Tree con tres hermanos y 4 puertos de servicio. Esta configuración requiere la reserva de 10 puertos del switch (1 puerto para la conexión con padre, 2 para los hijos, 3 para los hermanos y 4 para el servidor-proxy) para su implementación en cada nodo intermedio. El resto de puertos se pueden utilizar para la conexión de los usuarios.

La nueva topología de la arquitectura Double P-Tree provoca la aparición de lazos en el encaminamiento de la red. Este encaminamiento más complejo no puede ser resuelto por switches estándares, ya que estos últimos necesitan topologías sin lazos para poder redirigir correctamente los paquetes recibidos.

La arquitectura Double P-Tree requiere de componentes de red que puedan implementar políticas de encaminamiento de paquetes más inteligentes y flexibles que puedan trabajar con topologías con lazos. Dos alternativas que pueden cumplir estor requisitos son la utilización de routers ó la utilización de switches de nivel 3 en las redes locales [Cisc1] [Cisc00] [Swt1] [Net1].

3.2.2 Optimización de los recursos de red

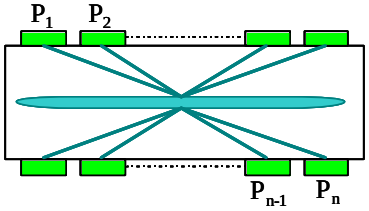
Una de las características más importantes de los sistemas totalmente distribuidos es que permiten que el tráfico de la red se pueda distribuir entre diferentes fuentes / puertos. Hasta ahora, siempre hemos asumido la utilización de redes no segmentados, las cuales no permiten aprovechar las características específicas de nuestras arquitecturas.

A continuación analizaremos la funcionalidad de los distintos tipos de redes (no segmentadas y segmentadas) para decidir cual de ellas se ajusta mejor a las características de las arquitecturas P-Tree.

En la figura 3-7, mostramos los requerimientos de las redes no segmentadas. En estas redes los requisitos de ancho de banda están definidos por la suma del tráfico de cada uno de los puertos.

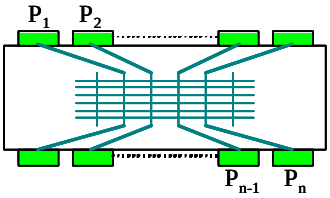
Las redes no segmentadas son apropiadas cuando el tráfico de la red es generado principalmente por una única fuente / puerto. Este modelo de tráfico es el que siguen los sistemas centralizados pero no coincide con el modelo de tráfico de los sistemas distribuidos.

En las redes segmentadas, cada uno de los puertos dispone de un ancho de banda independiente, tal y como se muestra en la figura 3-8. Los requisitos de ancho de banda de estas redes viene definido por el puerto del switch que más tráfico soporta. Estos switches son apropiados para redes en las cuales el tráfico soportado por la red se distribuye a través de diversos puertos. Las redes segmentadas permiten reducir el ancho de banda de red siempre que el tráfico entre los distintos puertos este equilibrado.



$$\text{Ancho de banda} = \sum_{i=1}^{n} Tráfico(P_i)$$

Figura 3-7. Requisitos de ancho de banda en las redes no segmentadas



$$\text{Ancho de banda} = Max_{i=1}^{n}\left[Tráfico\left(P_i\right)\right]$$

Figura 3-8. Requisitos de ancho de banda en las redes segmentadas

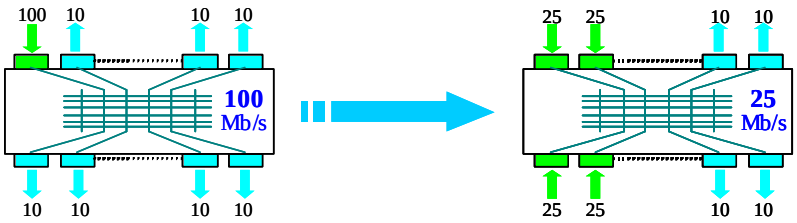
Utilización de múltiples puertos de servicio

Los beneficios asociados con la utilización de redes segmentadas se pueden ver mermadas por la características del tráfico en los sistemas de VoD. En estos sistemas existe un puerto que centraliza la mayor parte del tráfico que circula por la red: el puerto de servicio, al cual está conectado el servidor (ver figura 3-9a). Al ser este puerto el que tiene el mayor tráfico, es el que define los requisitos de ancho de banda de la red.

Para solventar este desequilibrio entre el puerto de servicio y el resto de puertos de la red, proponemos la conexión de los servidores al switch de la red mediante diversos puertos. La figura 3-9 muestra un ejemplo de bs beneficios asociados con este planteamiento en concreto y en general del potencial de las redes segmentadas para reducir los requisitos de ancho de banda de red del sistema.

En la figura 3-9a se muestra la distribución de tráfico en una red segmentada, en la cual el servidor utiliza un único puerto de servicio. En el ejemplo, este puerto recibe un tráfico de 100 Mb/s, considerablemente superior al tráfico recibido por el resto de puertos del switch, 10 Mb/s. En este caso, el ancho de banda requerido para la red seria de 100 Mb/s (el máximo tráfico de todos los puertos).

Estos altos requisitos para la red local son debidos a que el tráfico del servidor está centralizado en único puerto. Se puede reducir el ancho de banda de la red siempre que se pueda distribuir el tráfico del servidor entre diversos puertos.



(a) Configuración con un puerto de servicio **(b)** Configuración con 4 puertos de servicio

Figura 3-9. Distribución del tráfico del servidor local

En el ejemplo de la figura 3-9b, se utilizan 4 puertos de servicio para conectar el servidor a la red. Al distribuir la carga del servidor entre los 4 puertos, se reduce el tráfico recibido por cada uno de ellos a 25 Mb/s. Al tener tráfico del switch más balanceado, los requisitos de la red local se reducen en un 75% hasta los 25 Mb/s.

Esta misma técnica es la que aprovecha la arquitectura Double P-Tree para reducir los requerimientos de red del sistema, pero esta vez distribuyendo la carga del servidor local entre los servidores adyacentes, gracias a las peticiones servidas remotamente.

Utilizando redes no segmentadas el tráfico asociado con las peticiones remotas incrementan las necesidades de ancho de banda del sistema. Sin embargo, mediante la utilización de redes segmentadas se puede aprovechar del hecho que el tráfico remoto utilice puertos distintos del tráfico local para que el ancho de banda adicional requerido para servir las peticiones remotas no aumente los requisitos de las redes locales. Siempre que el tráfico asociado con las peticiones remotas no supere al tráfico de los puertos de servicio, no involucrarán un incremento en los requisitos de ancho de banda del sistema y se logrará un mejor rendimiento del sistema distribuido de VoD.

3.3 Modelo analítico de la arquitectura Double P-Tree.

Para evaluar la efectividad y la escalabilidad de la arquitectura Double P-Tree necesitamos adaptar el modelo analítico de la arquitectura P-Tree del capítulo anterior, teniendo en cuenta las nuevas características añadidas.

La escalabilidad del sistema se estima, evaluando el crecimiento del tráfico generado por el sistema y la evolución de los requisitos de ancho de banda de red para los componentes del sistema cuando éste crece. A diferencia de la arquitectura P-Tree, que todavía tiene reminiscencias de componentes centralizados, la arquitectura Double P-Tree está totalmente distribuida. Por lo tanto, para validar la escalabilidad de la nueva arquitectura se necesita verificar el crecimiento de los requisitos de todos los componentes de la topología (no solo los del primer nivel).

Durante el análisis, vamos a seguir utilizando el ancho de banda efectivo del sistema como principal métrica de rendimiento, aunque otras medidas tales como el ancho de banda de sistema, los requisitos de las redes locales, la capacidad de servicio y de almacenamiento de los servidores locales nos aportarán una valiosa información sobre los límites de la arquitectura respecto al número de usuarios soportados, su coste y grado de escalabilidad.

Al igual que en el modelo anterior, asumiremos una política de servicio unicast, que todos los niveles de la arquitectura están completos y que la frecuencia de acceso a los contenidos del catálogo sigue una distribución Zipf.

Para facilitar el seguimiento del modelo analítico, en la tabla 3-3 mostramos la notación que se va seguir durante el análisis.

Tabla 3-3. Notación del modelo analítico Double P-Tree

Símbolo	Definición	Símbolo	Definición
B_T	Ancho banda de red total	S	Número de puertos de servicio
B_e	Ancho banda efectivo del sistema	V	Número de videos en el catálogo
B_p	Ancho banda en la red principal	C	Tamaño de la cache en los servidores-proxy (en nº de videos)
B_c	Ancho banda de la redes locales	M	Tamaño del mirror en los servidores-proxy (en nº de videos)
B_u	Ancho banda efectivo de los usuarios	P_{mc}	Probabilidad de fallo en la cache en los servidores locales
L	Número de niveles de la topología	P_{hc}	Probabilidad de acierto en la cache en los servidores locales
N	Número de redes locales	P_{mm}	Probabilidad de fallo en cache+mirror en los servidores locales
O	Orden del árbol	P_{mc}	Probabilidad de acierto en la cache+mirror en los servidores locales
B	Número de hermanos		

Los principales cambios que se deben realizar al modelo analítico de la arquitectura P-Tree son debidos a la definitiva eliminación del servidor principal y de la red principal, a la utilización de redes segmentadas y a la modificación de la topología.

El ancho de banda efectivo (B_e) de la arquitectura Double P-Tree se puede calcular como la suma de los anchos de banda efectivos (B_u) para todas las redes locales, como:

$$B_e = B_u \cdot n \qquad (3.\mathbf{10})$$

A partir de este ancho de banda efectivo, podemos calcular los requisitos de ancho de banda de red total del sistema como la suma de los requisitos de red (Bc_i) de todas la redes locales del sistema.

Los requisitos de una red local depende de su posición en la topología. Debido a que la conectividad para todas las redes de un nivel es la misma, todas las redes locales de un nivel comparten los mismo requisitos. De esta forma el ancho de banda total del sistema (B_T) se puede evaluar en función del número de redes de cada nivel de la topología (o^i), multiplicado por los requisitos de las redes locales de ese nivel (Bc_i):

$$B_T = \sum_{i=0}^{L} o^i \cdot Bc_i \qquad (3.\mathbf{11})$$

Para calcular el ancho de banda requerido por una red del nivel *i* (*Bc*$_i$), hay que tener en cuenta el modelado de los requisitos específicos de las redes segmentadas. Tal y como ya hemos comentado en el apartado anterior, los requisitos de ancho de banda de una red segmentada esta definido por el puerto que tiene más tráfico.

Para evaluar los requisitos de una red local, debemos evaluar el tráfico que tiene que soportar cada uno de sus puertos y quedarnos con el máximo:

$$Bc_i = Max(Tp_i, Tf_i, Tc_i^1, \cdots, Tc_i^o, Th_i^1, \cdots, Th_i^b) \qquad (3.\mathbf{12})$$

En este caso, solo tendremos en cuenta el tráfico de los puertos de servicio (Tp_i) y los puertos de la topología ($Tf_i, Tc_i^1, \cdots, Tc_i^o, Th_i^1, \cdots, Th_i^b$). Los puertos de los usuarios son los menos saturados y en el análisis asumiremos que no influyen en los requerimientos finales de la red.

El tráfico gestionado por los puertos de servicio (*Tp*$_i$) esta generado por dos fuentes: por las peticiones locales (*Tpl*$_i$) y por las peticiones remotas (*Tpr*$_i$) recibidas desde el restos de la arquitectura. Además, si tenemos en cuenta que el servidor se puede conectar a la red a través de varios puertos (*S*) entonces el tráfico de cada uno de estos puertos se puede evaluar como:

$$Tp_i = \frac{Tpl_i + Tpr_i}{S} \qquad (3.\mathbf{13})$$

El volumen de tráfico asociado con las peticiones locales se puede calcular en función del porcentaje de acierto de la cache y del mirror del servidor local ($1 - p_{mm}(i,0)$), multiplicado por el tráfico generado por los usuarios locales (*Bu*), tal y como muestra la siguiente expresión:

$$Lps_i = Bu \cdot (1 - p_{mm}(i,0)) \qquad (3.\mathbf{14})$$

El tráfico recibido por los puertos de servicio debido a las peticiones remotas depende de la posición del servidor en la topología y se puede evaluar de la siguiente forma:

$$Lpr_i = Bu \cdot \sum_{d=1}^{L*2} Ne(i,d) \cdot \frac{p_{mm}(i,d)}{r_{\ln}(i,d)} \qquad (3.\mathbf{15})$$

El tráfico remoto recibido también se ve afectado por el número de servidores situados a una determinada distancia d de la red local del nivel *i* (*Ne(i,d)*). El trafico generado por estos

servidores a distancia *d*, se deben distribuir entre todos los servidores situados a esa misma distancia. Un porcentaje de estas peticiones (r_{ln}) serán atendidas por el servidor local *i*.

Para evaluar el porcentaje de peticiones que un pueden ser atendida por los servidores a distancia d, nos basamos en la expresión (3.7) del apartado 3.2.1, pero cambiando el valor aproximado para el número de nodos vecinos por la función Nd que devuelve el número de redes locales situadas a una distancia menor ó igual a d de una red en el nivel i de la topología:

$$p_{mm}(i,d) = \left(p_{hc} + p_{mc} \cdot \frac{M \cdot Nd(i,d)}{V - C} \right), p_{mm}(i,d) \ni [0,1] \qquad (3.\mathbf{16})$$

De forma similar el tráfico gestionado por los puertos de la topología (T_f, T_c y T_h) se puede evaluar como el volumen de tráfico asociado con las peticiones remotas atendidas por el servidor local (Tp_i), por el tráfico asociado con las peticiones que no pueden servirse localmente y tienen que ser gestionadas por servidores remotos (Tpr_i) y por último el tráfico en transito perteneciente a los streams que utilizan la red local como parte de su camino para llegar desde un servidor remoto a la red local del usuario.

Mientras que las dos primeras fuentes de tráfico se puede evaluar utilizando las expresiones (3.14) y (3.15), el cálculo del tráfico en transito es mucho más complicado de evaluar analíticamente ya que requiere conocer los caminos que van a seguir las peticiones mientras se sirven. Por esta razón, nos vamos apoyar en un simulador analítico que evalué las expresiones desarrolladas en este apartado y permita calcular la distribución exacta del tráfico entre los puertos de la topología de la redes locales.

3.4 Análisis del rendimiento de la arquitectura Double P-Tree

En este apartado analizaremos el rendimiento de la arquitectura Double P-Tree utilizando el modelo analítico previamente desarrollado. El principal objetivo de este análisis es demostrar que la Double P-Tree mejora la eficiencia de la arquitectura P-Tree, pero manteniendo una de las propiedades más importantes de ésta, la escalabilidad.

En la elaboración de este estudio asumiremos una configuración para el sistema LVoD por defecto. El sistema dispondrá de un ancho de banda de red total de 127.000 Mb/s. Salvo que se indique lo contrario se asumirá la utilización de redes segmentadas. Las arquitecturas distribuidas están compuestas por 255 redes locales con un ancho de banda de 500 Mb/s cada una. Esto implica a su vez que los sistemas P-Tree y Double P-Tree están compuestos por 8 niveles.

Los servidores-proxy dispondrán de una capacidad de almacenamiento suficiente para almacenar el 20% de los videos del catálogo del sistema. El aumento de la eficiencia de la

arquitectura Double P-Tree nos permite reducir volumen de almacenamiento requerido por los servidores-proxy, lo cual nos permite reducir estos requisitos del 30%-40% de la arquitectura P-Tree a únicamente el 20% con la arquitectura Double P-Tree.

Los servidores de la arquitectura estarán conectados a las redes locales mediante 1 y 4 puertos de servicio. La utilización de varios puertos de servicios (y el consecuente incremento del ancho de banda de servicio del sistema, que se multiplica también por 4) en esta experimentación se justifica también por el incremento de la eficiencia de arquitectura Double P-Tree y la utilización de redes segmentadas. Si sólo se utiliza un puerto de servicio la infraestructura de comunicaciones quedará infrautilizada y nos será imposible derivar conclusiones útiles del análisis de los resultados (todas las arquitecturas estudiadas obtendrían resultados similares). Por lo tanto, nos hemos decantado por incrementar la capacidad de servicio de los servidores, mediante el incremento del número de puertos de servicio y lograr, de esta forma, saturar la red de transmisión y obtener un mejor aprovechamiento de las redes del sistema LVoD.

3.4.1 Double P-Tree en contraposición a P-Tree

El análisis realizado de la arquitectura P-Tree en el capítulo anterior se fundamentó en la utilización redes no segmentadas. El primer paso en el análisis de la eficiencia de ambas arquitecturas consiste en comparar su rendimiento utilizando redes no segmentadas en ambas arquitecturas.

En la figura 3-10, mostramos el rendimiento de ambos sistemas cuando utilizan redes locales no segmentadas. Con estos resultados podemos comprobar que la arquitectura Double P-Tree consigue mejorar la eficiencia del sistema P-Tree en alrededor de un 38% con la configuración

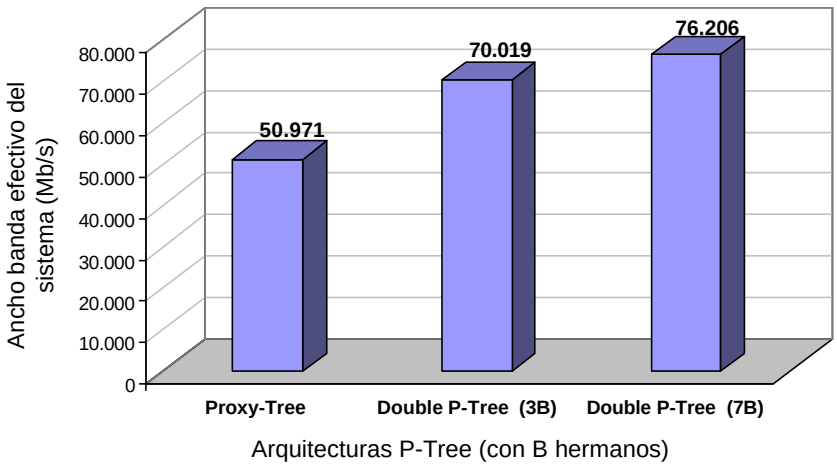


Figura 3-10. Rendimiento de las arquitecturas P-Tree con redes no segmentadas

con 3 servidores hermanos y en un 49% si se utiliza la configuración con 7 servidores hermanos.

Aunque la mayor eficiencia de la arquitectura Double P-Tree resulta evidente, su verdadero potencial se logra cuando se utilizan redes segmentadas. En la figura 3-11, se muestra de nuevo el ancho de banda efectivo para las dos arquitecturas, pero esta vez utilizando redes segmentadas y varios puertos de servicio para incrementar la efectividad del sistema.

Primeramente, queremos dejar constancia de la mejora de rendimiento obtenida por nuestras arquitecturas, con la única inclusión de las redes segmentadas. Si comparamos los resultados obtenidos por la arquitectura P-Tree utilizando redes no segmentas (fig. 3-10) y redes segmentadas (fig 3-11), podemos ver un incremento del rendimiento del sistema de casi el 150% (126.873 Mb/s contra 50.971 Mb/s). Estos resultados obtenidos con la participación de las redes segmentadas esta muy cercanos al rendimiento óptimo (127.000 Mb/s, dada la capacidad de servicio de los servidores utilizados) que se puede obtener con dicha configuración del sistema.

Al comparar los resultados por las dos arquitecturas utilizando redes segmentadas podemos comprobar al utilizar un único puerto de servicio que los rendimientos son muy parecidos (126.873 Mb/s contra 127.000 Mb/s). Esto es debido a que en ambas, la configuración utilizada está limitada por el ancho de banda de servicio y únicamente permite alcanzar un ancho de banda efectivo máximo de 127.000 Mb/s.

Si incrementamos la capacidad de servicio de los servidores locales del sistema, incrementando los puertos de servicio hasta 4 (cada uno de ellos puede alcanzar un ancho de banda de servicio de 1.000 Mb/s), podemos contrastar la desigual eficiencia de ambos sistemas. La arquitectura Double P-Tree con 3 servidores hermanos y 4 puertos de servicio obtiene un ancho de banda efectivo de 388.000 Mb/s, casi dos veces y medio superior al rendimiento obtenido por la arquitectura P-Tree utilizando con la misma configuración (157.000 Mb/s).

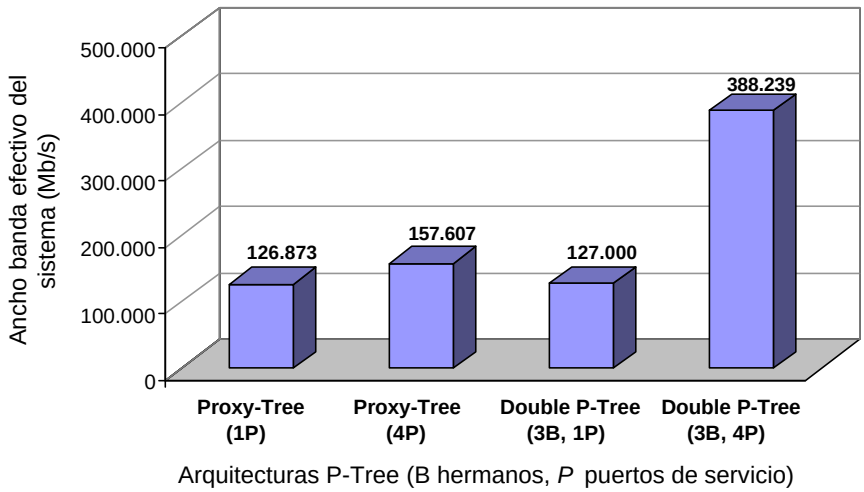


Figura 3-11. Rendimiento de las arquitecturas P-Tree utilizando redes segmentadas

3.4.2 Escalabilidad

El análisis de la escalabilidad para la arquitectura Double P-Tree difiere ligeramente del realizado para la arquitectura P-Tree. En el sistema Double P-Tree ya no existe un componente central (servidor / red principal) que se tenga que controlar especialmente para garantizar la escalabilidad del sistema. En cambio, al tratarse de una arquitectura totalmente distribuida, para verificar que el sistema puede escalar sin causar saturación hay que comprobar que los requisitos de ancho de banda de cada uno de los componentes están acotados.

La escalabilidad del sistema la podemos observar mediante la figura 3-12, que muestra el máximo ancho de banda requerido por cualquiera de las redes locales (grafica 3) del sistema y el máximo ancho de banda requerido por cualquiera de los puertos de servicio de la arquitectura (grafica 4), a medida que el sistema escala. Estos resultados se han obtenido aplicando las expresiones (3.10) y (3.11) para calcular el ancho de banda efectivo del sistema y los requisitos de ancho de banda las redes a medida que se incrementa el número de redes locales (*N*).

En la figura observamos que incluso incrementando exponencialmente la capacidad de servicio del sistema (gráfica 4) el ancho de banda requerido por las redes locales se mantiene acotado en los 1.200 Mb/s. Lo mismo ocurre con los requisitos de ancho de banda de servicio en los puertos de los servidores locales, el cual no supera los 400 Mb/s (si tenemos en cuenta que cada servidor local utiliza 4 puertos de servicio, la capacidad máxima de los servidores locales será de 1600 Mb/s).

Estos resultados nos permiten concluir, sin lugar a dudas, que la arquitectura Double P-Tree permite una escalabilidad ilimitada, utilizando componentes pequeños.

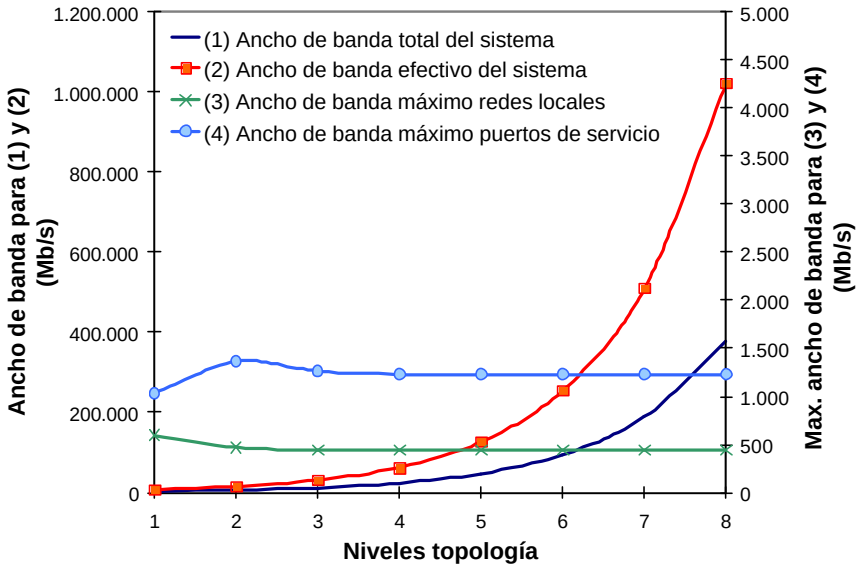


Figura 3-12. Escalabilidad arquitectura Double P-Tree

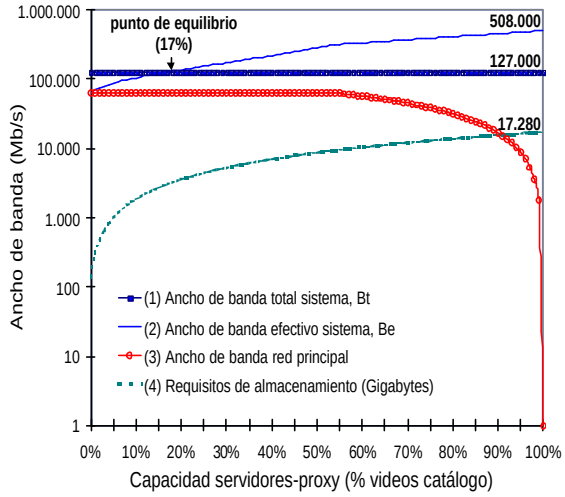
3.4.3 Requerimientos de almacenamiento en los servidores-proxy

En este apartado analizaremos la eficiencia del almacenamiento de los servidores locales para las arquitecturas de un nivel de servidores-proxy y Double P-Tree. Para ambas arquitecturas mostramos el rendimiento del sistema (ancho de banda efectivo) y los requisitos de almacenamiento a medida que se incrementa la capacidad de almacenamiento de los servidores.

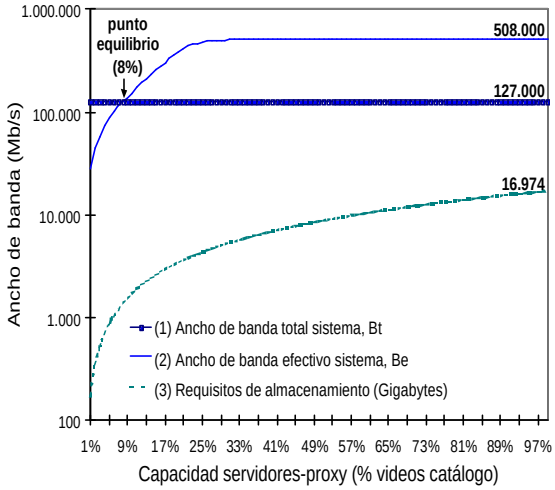
En la figura 3-13a, podemos evaluar la eficiencia del almacenamiento en la arquitectura de servidores-proxy de un nivel. Mientras que con la figura 3-13b se representa la efectividad del almacenamiento en la arquitectura Double P-Tree.

En la misma gráfica (3-13a) queda reflejado el comportamiento de las tres arquitecturas: En el extremo izquierdo, cuando la capacidad de los servidores-proxy es nula, tenemos una arquitectura que se comporta como un sistema centralizado basado en dos niveles de redes. En el extremo derecho tenemos un sistema de servidores independientes, ya que los servidores-proxy tienen suficiente capacidad para almacenar todo el catálogo. Entre ambos extremos se refleja la eficiencia de un sistema de servidores-proxy de un nivel con diferentes capacidades de almacenamiento.

Estos resultados permiten constatar como **la arquitectura de servidores-proxy de un nivel surge como un compromiso entre los sistemas centralizados** (menos almacenamiento y menos escalabilidad) **y los sistemas de redes independientes** (más requisitos de almacenamiento y mayor escalabilidad).



(a) Un nivel de servidores-proxy



(b) Double P-Tree

Figura 3-13. Eficiencia del almacenamiento en sistemas basados en servidores-proxy

Comparando ambas arquitecturas basadas en servidores-proxy, podemos ver que el sistema **Double P-Tree obtiene un mejor rendimiento de la capacidad de almacenamiento de los servidores locales**. Esta afirmación queda demostrada si comparamos la pendiente de crecimiento del ancho de banda efectivo del sistema de ambas arquitecturas a medida que se aumenta la capacidad de los servidores locales / servidores-proxy.

El rendimiento de la arquitectura Double P-Tree progresa más rápido, alcanzando la cota óptima (508.000 Mb/s, 127.000 Mb/s * 4 puertos de servicio) necesitando únicamente una capacidad en lo servidores locales del 25%. En este punto, el tráfico requerido en los puertos de la topología para atender las peticiones remotas es menor que el ancho de banda requerido en lo puertos de servicio y por lo tanto, ya no imponen una penalización sobre el rendimiento del sistema. A partir de esta capacidad, el rendimiento del sistema no aumenta ya que se encuentra limitado por la capacidad de servicio de los servidores que esta limitada a 508.000 Mb/s.

Mientras tanto se puede observar que el crecimiento del rendimiento en la arquitectura de un nivel de servidores-proxy es menos dinámico, no alcanzando su cota máxima hasta que los servidores-proxy pueden almacenar todo el catálogo. Esta menor eficiencia de la capacidad de los servidores-proxy es debido a que los enlaces de conexión de las redes locales con el servidor principal se convierten en el cuello de botella de este sistema.

Continuando con la comparación de ambos sistemas, se puede observar que el punto de equilibrio (el punto en que el ancho de banda efectivo del sistema coincide con el ancho de banda físico) se alcanza en la arquitectura Double P-Tree con una capacidad de almacenamiento del 8%, mientras que la arquitectura de servidores-proxy de un nivel necesita una capacidad del 17% (más del doble) para alcanzar el mismo objetivo.

A partir de este estudio, podemos concluir que la arquitectura Double P-Tree es un sistema mucho más eficiente que la arquitectura de servidores-proxy de un nivel.

3.4.4 Comparación de resultados de la arquitectura Double P-Tree con otras alternativas LVoD

Al igual que hicimos con la arquitectura P-Tree (apartado 2.5.3 del capítulo anterior), ahora procederemos a comparar la arquitectura Double P-Tree con los principales aproximaciones existentes para la construcción de sistemas LVoD escalables (servidores independientes y servidores-proxy de un nivel). También nos compararemos con una arquitectura centralizada, pero solo se utilizará como referencia, debido a que su nula escalabilidad y altos costes no son consistentes con los objetivos de diseño para un sistema LVoD.

La arquitectura de servidores-proxy de un nivel, al igual que ocurrió en el capítulo anterior, se convierte en nuestra principal referencia en este estudio debido a la similitud entre los conceptos de diseño y los recursos utilizados. En la figura 3-14 podemos observar que Double P-Tree mejora en todo momento el rendimiento obtenido con la arquitectura de servidores-proxy de un nivel. Utilizando una configuración con 7 hermanos, Double P-Tree supera en un 350% el ancho de banda efectivo de la arquitectura de servidores-proxy de un nivel.

Además de este mejor rendimiento, también podemos comprobar como la red y los servidores con mayores requisitos del sistema Double P-Tree son 15 veces más pequeños que el servidor central ó la red principal de la arquitectura de servidores-proxy de un nivel.

Si ahora nos fijamos en los resultados de la arquitectura de servidores independientes, podemos observar que esta arquitectura logra un incremento del rendimiento del 4% respecto a nuestra propuesta (508.000 Mb/s contra 490.000 Mb/s). Este rendimiento menor puede ser totalmente subsanado incrementando ligeramente la capacidad de almacenamiento de los servidores hasta el 25%.

También estamos convencidos que este pequeño hueco en el rendimiento puede ser corregido y superado ampliamente por la arquitectura Double P-Tree cuando se tenga en cuenta el efecto de las políticas multicast (indispensables para el redimiendo de los sistemas LVoD) sobre la eficiencia del sistema. Nuestro optimismo se debe al hecho de que la eficiencia de las técnicas multicast dependen en gran medida del número de usuarios que pueden ser atendidos por los servidores. En las arquitecturas de servidores independientes éste número está limitado por los usuarios conectados a la red local, limitando la capacidad de compartición de recursos (ancho de banda de red y ancho de banda de servicio) exclusivamente dentro del rango de cada

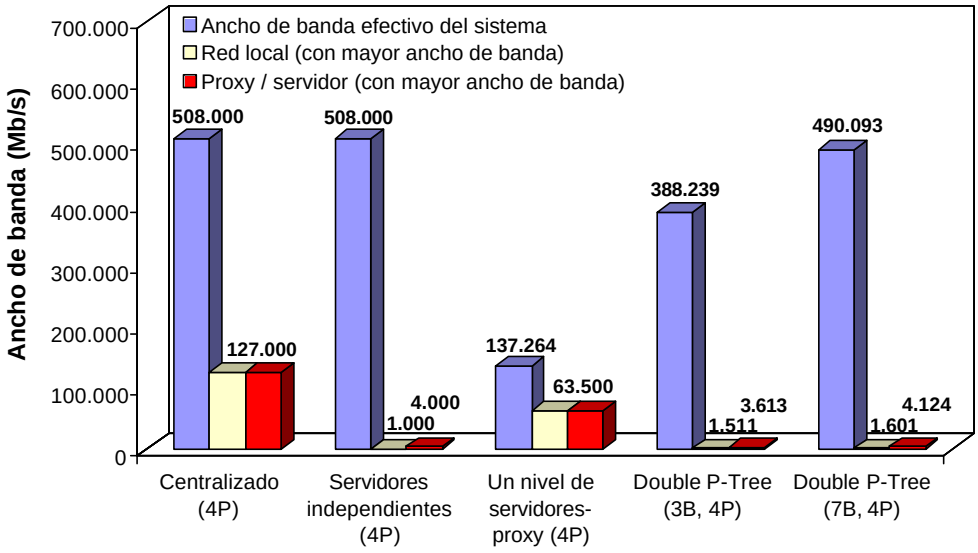


Figura 3-14. Rendimiento y requisitos de ancho de banda de las arquitecturas LVoD

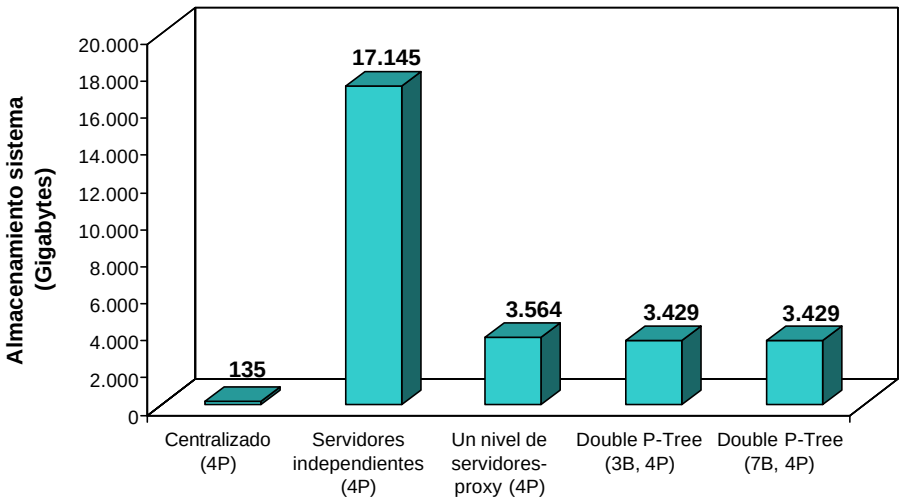


Figura 3-15. Requisitos de almacenamiento de las arquitecturas LVoD

red local. Mientras tanto, en Double P-Tree, los servidores locales pueden gestionar peticiones de las redes locales adyacentes, incrementando con ello el número de usuarios que pueden compartir los recursos del sistema. En una configuración con 4 hermanos, la probabilidad de compartición puede llegar a ser 6 veces mayor (conectividad media de la topología) que en el sistema de servidores independientes.

La figura 3-15, muestran los requisitos de almacenamiento asociado con cada una de las arquitecturas analizadas. Podemos comprobar que las arquitecturas basadas en servidores-proxy tienen unos requisitos similares, alrededor de los 3.500 GBytes. Sin embargo, la arquitectura de servidores independientes necesita 5 veces más almacenamiento (17.145 GBytes contra 3.429 GBytes) que la arquitectura Double P-Tree. Estos mayores requisitos de la arquitectura de servidores independientes no se justifican con la diferencia de rendimiento (de apenas el 4%) obtenida respecto a la arquitectura Double P-Tree.

3.5 Conclusiones

En este capítulo hemos presentado una nueva arquitectura de VoD distribuida a gran escala denominada Double P-Tree. Esta arquitectura está basada en la arquitectura previa Proxy-Tree y tiene por objetivo mejorar su eficiencia.

Un concepto clave en el diseño de esta nueva arquitectura es la distancia media de servicio. Esta distancia refleja el número de redes del sistema que deben cruzar los datos enviados por el servidor hasta llegar al usuario. Para un sistema distribuido, la distancia media de servicio

depende del almacenamiento disponible en cada uno de los nodos distribuidos de servicio, de los esquemas bajo los cuales se gestiona este almacenamiento y sobre todo de la conectividad de la topología subyacente a la arquitectura del sistema VoD.

Para mejorar la eficiencia se han llevado a cabo diversas modificaciones en la arquitectura:

- Para lograr un sistema completamente distribuido se ha eliminado el servidor central y la red principal, distribuyendo su funcionalidad entre todos los servidores locales.
- Para incrementar la efectividad del sistema se ha incrementado la conectividad de la topología utilizando el concepto de servidores hermanos.
- Por último, para aprovechar mejor las características de la arquitectura distribuida y reducir los requisitos de red del sistema se ha propuesto la utilización de switches segmentados en las redes locales.

Gracias a estas modificaciones la arquitectura Double P-Tree ha alcanzado una mayor eficiencia, mejorando el rendimiento de la arquitectura P-Tree en un 210% y del sistema de un nivel de servidores-proxy en un 350%. Double P-Tree también ha demostrado ser capaz de alcanzar una efectividad similar a la arquitectura de servidores independientes, con unos requisitos de almacenamiento 5 veces menores.

Otra de las características que se mejora en la nueva arquitectura es la tolerancia a fallos. El incremento de la conectividad y la no dependencia de un servidor central ha incrementado el número de componentes que pueden fallar antes de que el servicio en el sistema pueda quedar interrumpido. Un aspecto importante estriba en que estas características se han podido alcanzar conservando la ilimitada escalabilidad de la arquitectura original.

Sin embargo, estas mejoras no se han podido alcanzar sin costes adicionales. La nueva topología puede provocar la aparición de lazos en las comunicaciones del sistema, lo cual requiere la utilización de redes con capacidades de encaminamiento de los paquetes en las redes locales.

La capacidad de mejora de la arquitectura Double P-Tree con respecto a otros sistemas, no se ha agotado con las mejoras presentadas. Dada la alta conectividad de nuestra arquitectura, creemos que la intervención de las políticas de multicast puede permitir que Double P-Tree pueda demostrar todo su potencial. Por lo tanto, nuestro siguiente objetivo se dirige hacia el análisis de la efectividad de las políticas multicast y el estudio de la políticas de gestión a bajo nivel (distribución del almacenamiento, políticas de mapping de contenidos, etc...) de la arquitectura Double P-Tree.

Dado el nivel de complejidad ya alcanzado por lo modelos analíticos propuestos y la complejidad de modelar las nuevas políticas que se quieren estudiar, hace inviable continuar el

estudio de la arquitectura utilizando exclusivamente modelos analíticos. Por esta razones y debido a que no existen en la literatura simuladores con la funcionalidad requerida, hemos optado por el desarrollo y la utilización de un simulador de sistemas de VoD para poder continuar con el análisis de las arquitecturas P-Tree.

En el siguiente capítulo vamos a presentar el diseño y la funcionalidad principal de este simulador y su validación utilizando los resultados de los modelos analíticos.

Capítulo 4

VoDSim, una herramienta para la evaluación de arquitecturas LVoD

Resumen

Este capítulo se presenta el diseño y el modelado de la herramienta VoDSim para la simulación de las arquitecturas LVoD. Se especifica la funcionalidad requerida para el simulador, las características soportadas, el modelo y los parámetros de configuración soportados. Finalmente, se demuestra la bondad de los resultados alcanzados mediante la herramienta de simulación validándolos respecto a los resultados obtenidos con los modelos analíticos propuestos en los capítulos segundo y tercero.

4.1 Introducción

La simulación consiste en el uso de los ordenadores para duplicar o imitar el comportamiento de un determinado sistema. Estos sistemas pueden existir físicamente o ser completamente ficticios. En ambos casos el objetivo de la simulación es permitir realizar conjeturas sobre el modelo, que se puedan aplicar al sistema sin necesidad de manipular o modificar el sistema real.

La simulación como herramienta de análisis consta de una serie de características que la hacen muy interesante para su utilización:

- Gran flexibilidad.

  La simulación permite modificar fácilmente diversos parámetros del sistema para su posterior estudio. Por ejemplo se puede expandir el tiempo, permitiendo la obtención de estadísticas de interés en pequeños intervalos de tiempo simulado. Esto permite el estudio de la estructura detallada de los cambios en el sistema, el cual posiblemente no se pueda observar en el sistema real.

  A través de la simulación se pueden estudiar los efectos de ciertos cambios sobre la operación de un sistema, realizando alteraciones en el modelo y observando su efecto sobre el comportamiento del sistema modelado.

- Eficiencia.

  La simulación puede comprimir el tiempo de manera que largos intervalos de tiempo sobre el sistema real, requieren pocos segundos sobre el modelo simulado. Esto permite ahorrar mucho tiempo durante el análisis de los sistemas reales.

- Aislamiento con respecto al sistema físico.

  Hace posible el estudio y experimentación con complejas iteraciones de un sistema dado o parte del mismo sin afectar al sistema real.

- Mejora la comprensión del sistema real.

  Una detallada observación del sistema simulado puede conducir a un mejor conocimiento del mismo y provocar mejoras sobre el sistema real que de otra manera no serían visibles. Además la experiencia de diseñar un modelo de simulación puede ser más beneficiosa que la simulación es si misma. El conocimiento obtenido al diseñar un estudio de simulación frecuentemente sugiere cambios en el sistema que esta siendo simulado. Los efectos de estos cambios pueden ser comprobados vía simulación antes de ser implementados sobre el sistema real.

La simulación de sistemas complejos también puede aportar un conocimiento acerca de cuáles son las variables y los parámetros más importantes del sistema y cómo éstas se relacionan entre si.

- Experimentación con el sistema físico.
  La simulación puede ser utilizada para experimentar con nuevas situaciones sobre las cuales existe poca o ninguna información, verificando que ocurre con el sistema real. Cuando nuevos componentes son introducidos en un sistema, la simulación puede ayudar a encontrar zonas de conflicto u otros problemas que comprometerían la operación del sistema real.

Las herramientas de simulación dentro del ámbito de los sistemas LVoD pueden cubrir 2 grandes objetivos: la investigación y la configuración / sintonización de los sistemas reales.

4.1.1 Investigación

Los sistemas de LVoD son muy costosos y requieren una amplia infraestructura. Por lo tanto, la investigación realizada sobre estos sistemas raramente pueden basarse en el sistema real.

Una alternativa muy utilizada para el análisis de estos sistemas es la utilización de modelos analíticos. Sin embargo su alta complejidad, limita su ámbito de actuación a aspectos muy concretos y acotados de la funcionalidad del sistema (consumo de ancho de anda, etc...). Estas herramientas de análisis no permiten estudiar la interacción de las diversas políticas / recursos dentro del sistema VoD ya que el modelo resultante seria demasiado complejo.

La simulación, por su parte, permite estudiar el sistema analizado en su conjunto o por partes. Esta característica permite analizar la integración de los distintos componentes del sistema o bien enfocarse en el examen de una política ó componente en concreto.

Las principales ventajas que nos aportan las herramientas de simulación a la investigación son:

1. Permitir analizar una gran variedad de sistemas LVoD.

   La simulación nos va a permitir estudiar en profundidad los parámetros de rendimiento de las distintas arquitecturas LVoD sin tener que implementarlos físicamente.

2. Nivel de abstracción.

   La simulación nos permite utilizar diferentes niveles de abstracción en los componentes del modelo del sistema físico.

Ciertos componentes del sistema pueden ser modeladas a muy bajo nivel de forma que se permita analizar cualquier aspecto relevante para la investigación. Al mismo tiempo, otros componentes ó políticas, que no interesan directamente para la investigación, se pueden abstraer completamente con el objetivo de simplificar el modelo del sistema.

Estas propiedad, nos ha permitido incrementar el nivel de detalle de aquellas políticas / componentes críticas (distribución del tráfico en las redes, saturación de los servidores, asignación de los videos a los proxies, ect..) en nuestra investigación del sistema. El aumento del nivel de detalle de determinadas políticas puede permitir incluso, su integración en el sistema real sin requerir grandes cambios. En el otro extremo, ciertas funcionalidades del sistema que no son imprescindibles desde el punto de vista de nuestra investigación se han simplificado lo máximo posible.

3. Flexibilidad en la ampliación del sistema.

   Es muy importante que el sistema simulado pueda ser ampliado y modificado de forma fácil para permitir analizar distintas configuraciones del mismo ó añadir nuevas políticas.

   Modificaciones que son muy difíciles de realizar sobre un sistema real sin afectar su funcionamiento normal, se pueden realizar sin embargo sin muchos problemas sobre el sistema simulado.

4.1.2 Sintonización y configuración de sistemas de VoD

Otra función que pueden desempeñar las herramientas de simulación es la de servir de apoyo para la configuración y sintonización de los sistemas de VoD reales.

La configuración óptima de un sistema LVoD no es fácil de obtener debido a que intervienen múltiples parámetros que se deben considerar.

Por ejemplo, uno de los problemas más complejos a resolver en la implementación es definir la capacidad del sistema. Los sistemas de VoD suelen estar diseñados para soportar solo una porción del volumen máximo de peticiones que se puede producir. Diseñar un sistema capaz de soportar simultáneamente todos los usuarios puede resultar prohibitivo desde el punto de vista del coste y altamente ineficiente ya que el sistema sufrirá una gran infrautilización durante la mayor parte del su tiempo de vida.

Ajustar un sistema de VoD a la carga que debe soportar de forma que se minimice el número de peticiones denegadas, requiere de un análisis previo a la propia construcción del sistema. Este análisis requiere tener en cuenta los distintos volúmenes de carga que puede soportar el

sistema (en función del horario y comportamiento de los usuarios) y, por lo tanto, hace imprescindible la utilización de una herramienta de simulación flexible.

Otros parámetros de configuración que pueden requerir la utilización de herramientas de simulación son:

- Dimensión ideal del sistema LVoD. Número de servidores y redes, y su tamaño.
- Capacidad de almacenamiento de los servidores locales.
- Definición de número de usuarios que puede soportar cada red del sistema.
- Configuración óptima de los contenidos almacenados en los servidores locales.
- Análisis del comportamiento del sistema frente a los picos de demanda.
- Planificación a medio / largo plazo del servicio de las peticiones.
- Detección de futuros cuellos de botella en el sistema.

Además, durante el funcionamiento del sistema real, la simulación puede servir de apoyo a las distintas políticas de gestión. Por ejemplo, la planificación de los recursos (analizando el rendimiento del sistema en función de las decisiones de planificación tomadas) ó en la redistribución de la carga del sistema en el caso de fallo en alguno de los componentes.

4.2 Diseño del simulador

Actualmente no existen simuladores de arquitecturas LVoD disponibles que modelen la funcionalidad (arquitecturas y parámetros) requerida por nuestra investigación. Por lo tanto, hemos tenido que diseña e implementar nuestro propio simulador para lograr los objetivos anteriormente descritos. Para acometer este propósito, ha sido de gran ayuda la amplia experiencia previa en el diseño e implementación de simuladores de eventos discretos, tanto secuenciales como distribuidos [Sup00][Sup00b][Sup01].

4.2.1 Requisitos del simulador

El principal objetivo del simulador es poder analizar y evaluar las distintas arquitectura de VoD propuestas y permitir el comparar su rendimiento con otras aproximaciones diferentes. El simulador debe soportar como mínimo las arquitecturas Proxy-Tree, Double P-tree y sus diferentes variantes (con ó sin servidor principal, con distintas topologías, etc...), así como las diferentes arquitecturas LVoD utilizadas como referencia: centralizada, de servidores independientes y un nivel de servidores-proxy.

Dada la importancia para los sistemas de VoD de que los distintos usuarios puedan compartir recursos entre si, el simulador deberá soportar tanto técnicas unicast, como técnicas multicast. Las técnicas unicast permiten evaluar la capacidad base de servicio de un sistema de VoD, mientas que las técnicas de multicast permiten la optimización de estos recursos base. De estas últimas hemos seleccionado las técnicas de batching y patching por ser, en nuestra opinión, las más significativas.

Para poder evaluar la efectividad de los sistemas VoD, el simulador debe admitir las distintas políticas de gestión empleadas en las distintas arquitecturas. En el caso de las arquitecturas P-Tree se deben soportar los diferentes esquemas de gestión utilizados, mirroring y caching.

El simulador también debe tener en cuenta el comportamiento del usuario, el cual puede afectar considerablemente al funcionamiento del sistema. El simulador debe ser capaz de soportar distintas cargas de trabajo (frecuencia de peticiones), así como el ciclo de comportamiento básico de un usuario dentro del sistema.

Una de las características de los simuladores utilizados en la investigación es su alta frecuencia de modificación y ampliación. Debemos dotar a nuestro simulador de una gran flexibilidad, de forma que permita la incorporación de nuevas características ó la modificación de las ya existentes de forma fácil y sin necesidad de rehacer la mayor parte de la funcionalidad ya implementada.

Otro aspecto importante en la simulación es la obtención de resultados y estadísticas del sistema simulado. En nuestro caso, nos interesa profundizar en el análisis de la evolución del ancho de banda de los componentes del sistema (servidores y redes) de forma que podamos estudiar su escalabilidad y eficiencia. Uno de los aspectos más importantes de los sistemas de LVoD, es la distribución del tráfico entre las redes del sistema y los puertos de éstas. El simulador debe ser capaz de generar todas estas estadísticas.

Por último, para que los resultados obtenidos por el simulador sean representativos, es necesario validar sus resultados contra los obtenidos por el sistema real ó el modelo analítico en el caso de que el sistema real no esté disponible. Esta validación se llevará a cabo en el último apartado de este capítulo.

4.2.2 Estructura del simulador

La figura 4-1 muestra la estructura básica del simulador diseñado y los distintos módulos que lo componen.

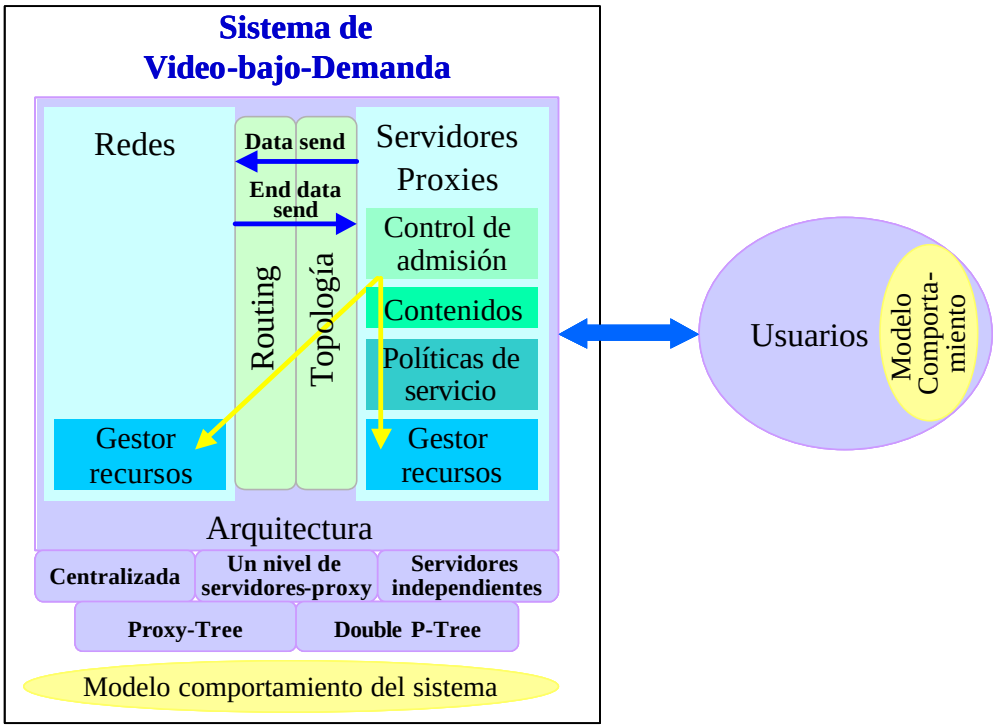


Figura 4-1. Estructura y módulos del simulador VoDSim para sistemas de VoD

El primer nivel dentro de la estructura jerárquica del simulador esta compuesto por la arquitectura del sistema de VoD. La arquitectura esta definida como un conjunto de servidores y redes interrelacionados entre si. El tipo de arquitectura y de los parámetros de configuración determinará la topología y el *routing* del sistema, que a su vez definirá la correspondencia existente entre los distintos componentes de la arquitectura.

Para especificar la funcionalidad de las redes y los servidores se han definido diversos módulos que permiten detallar los recursos disponibles y las políticas utilizadas para su gestión. Las principales políticas definidas dentro de los componentes de la arquitectura son las políticas de control de admisión, las políticas de servicio de peticiones, las políticas gestión de contenidos y las políticas gestión de recursos.

El módulo de control de admisión, implementa las políticas de admisión de peticiones. Este módulo se localiza dentro de cada uno de los servidores que componen la arquitectura. Su principal función es verificar que estén disponibles todos los recursos necesarios para poder atender una petición de un usuario. Este módulo está fuertemente acoplado con el resto de módulos del servidor y la red ya que requiere de información de éstos (disponibilidad ó no del contenido requerido, política de servicio utilizada y si hay suficientes recurso de servicio y de red para servir las peticiones).

El módulo de gestión de contenidos se encarga de gestionar los esquemas de cache y mirroring dentro de los servidores-proxy, así como de las distintas políticas de asignación / inicialización de contenidos.

Las políticas de servicio especifican como se va a servir una petición, en función de las distintas políticas activas en el servidor (unicast, batching, patching, etc...) Este módulo en conjunción con el módulo de control de admisión se encargan de reservar todos los recursos requeridos para servir la petición (ancho de banda de servicio y ancho de banda de red).

El último módulo integrado dentro de los servidores, y que también se localiza en los componentes de red, es el módulo de gestión de los recursos. Este módulo lleva el control de los recursos disponible en cada uno de los componentes del sistema, sirviendo de apoyo para las políticas de control de admisión y el mantenimiento del estado del sistema de VoD.

Además, en el simulador se incorporan dos módulos adicionales que modelan el comportamiento del sistema y de los usuarios. El modelo de comportamiento del sistema sirve para especificar distintas características que tienen validez en el ámbito global a todo el sistema. Entre estas características podemos destacar el número y las propiedades (tamaño, popularidad, etc...) de los contenidos que conforman el catálogo del sistema, número de usuarios y su distribución entre los distintos servidores / redes y el intervalo de llegada entre las peticiones de los usuarios.

El módulo de los usuarios define las características del STB mediante el cual se conectan los usuarios al sistema y el comportamiento de éste dentro del mismo.

En el siguiente apartado se definirá con más detalle la funcionalidad de cada uno de estos componentes.

4.2.3 Funcionalidad de los componentes del simulador

A continuación definiremos de forma más especifica, la funcionalidad concreta que debe soportar el simulador para cada uno de los componentes que integran un sistema de VoD.

4.2.3.1 Arquitecturas LVoD

El módulo de la arquitectura debe permitir soportar cualquiera de las configuraciones estudiadas a lo largo de esta tesis, por lo tanto se debe definir lo más genérico posible.

Inicialmente el simulador debe soportar múltiples arquitecturas, cada una de ellas con sus propias características. Básicamente, todas las arquitecturas admitidas pueden definir su funcionalidad especificando el tipo de arquitectura, el número y las características de los

elementos que la conforman (servidores / proxies, redes) y como están conectados entre si estos componentes.

El tipo de arquitectura nos va a definir una serie de parámetros representativos de su funcionalidad. Los principales parámetros que se pueden deducir de esta forma son la topología, el encaminamiento de la mensajes en la misma (routing) y la jerarquía de servicio existente entre los servidores de la arquitectura.

La topología de la arquitectura se puede definir de forma implícita en función del tipo arquitectura y su configuración (número de redes, orden del árbol, número de hermanos, etc...) ó bien especificando explícitamente cada uno de los enlaces de interconexión entre las distintas redes del sistema.

Una vez definida la topología el simulador puede inferir el routing a utilizar en la misma. El rounting es importante en el simulador debido a que a partir de él se calcula el camino que deben seguir cada uno de los streams en su transmisión desde el servidor a los usuarios. Este camino permite el cálculo de distancias entre los nodos de la arquitectura, permite la reserva de recursos de ancho de banda en cada una de las redes que se deben cruzar y calcular la latencia total de cada transmisión.

En sistemas distribuidos, la jerarquía de servicio especifica el orden en que los servidores gestionan las peticiones de los clientes. Esta jerarquía solo tiene sentido cuando trabajamos con arquitecturas en las cuales la atención al cliente está distribuida y jerarquizada (por ejemplo, sistemas basados en servidores-proxy).

La jerarquía de servicio dentro de la arquitectura de proxies de un nivel es la siguiente: primero se atiende la petición en los servidores-proxy locales y si éstos no pueden servirla, se recurre al servidor central. En caso de que este último tampoco pueda atender la petición, se deniega el servicio al usuario.

En la arquitecturas P-Tree los distintos niveles de la jerarquía de servicio están formados por el servidor local, los servidores remotos a distancia 2, los servidores remotos a distancia 3, y así sucesivamente hasta alcanzar los servidores remotos a distancia M (siendo M la máxima distancia entre dos puntos de la topología). La petición del usuario se denegará solo en el caso que ninguno de los servidores del sistema tenga suficientes recursos para atenderla.

Servidor / proxy

Los servidores son uno de los componentes clave en el modelado de un sistema de VoD. Para modelar un servidor de VoD es necesario especificar los recursos disponibles y su funcionalidad. Esta última se basa en las distintas políticas implementadas internamente, entre

las que destacamos entre las más relevantes: las políticas de control de recursos, la política de admisión de peticiones, las políticas de servicio y las políticas de gestión de contenidos.

Las arquitecturas simuladas soportan dos tipos de servidores: servidores estándar y servidores-proxy. Como ya se ha dicho, la principal diferencia entre ambos estriba en la capacidad de almacenamientos de contenidos y la utilización de ciertas políticas de gestión especificas (políticas de caching, mirroring, etc...).

Se supone que un servidor estándar tiene suficiente capacidad para almacenar todos los contenidos del sistema. Sin embargo, los servidores-proxy tienen una capacidad limitada, que normalmente no les permiten almacenar todos los videos del catálogo.

Los servidores-proxy deben modelar esta característica, lo cual implica asignarles una capacidad limitada (que repercute en los contenidos disponibles en los servidores-proxy) y llevar un control de los contenidos presentes en todo momento en el almacenamiento local.

En el caso de las arquitecturas P-Tree, además se debe tener en cuenta la distribución del almacenamiento del servidor-proxy entre cada uno de los esquemas de gestión de contenidos (caching y mirroring).

Redes de comunicación

Los principales parámetros que se requieren para simular una red de transmisión es el ancho de banda disponible (que nos permitirá caracterizar el volumen de tráfico que se puede soportar), el número de puertos de los switches, la latencia y el tipo de red.

El simulador soporta dos tipos diferentes de redes: las redes segmentadas y las redes no segmentadas. Como se ha comentado, la principal diferencia existente entre estos dos tipos estriba en la gestión de los recursos de la red (ancho de banda). Las redes no segmentadas se asumirá que el ancho de banda de red es único y se gestionará de forma global a la red. En las redes segmentadas se asumirá que cada puerto dispone de un ancho de banda independiente.

Otra aspecto en el cual difieren el modelado de ambas redes afecta al consumo de ancho de banda asociado con cada paquete que se transmite. Las redes no segmentadas gestionan el ancho de banda como un recurso global, por lo tanto, la transmisión de un mensaje sólo requiere una única reserva de ancho de banda. Por contra, cada puerto de una red segmentadas gestiona su ancho de banda de forma independiente y, por lo tanto, la transmisión de un mensaje requiere la realización de dos reservas de ancho de banda (una en el puerto de entrada del mensaje en la red y otra en el puerto de salida).

4.2.3.2 Políticas de gestión

Las políticas de gestión caracterizan una funcionalidad concreta dentro de los componentes que integran el sistema de VoD.

Políticas de control de admisión de peticiones

Para simular la política de control de admisión se utilizá una política básica que consiste en verificar si actualmente existen los recursos necesarios (en la red y en el servidor) para poder servir la petición. Inicialmente no se tendrá en cuenta (y por lo tanto no se reservarán recursos) para aquellas peticiones que siendo servidas mediante un canal de multicast abandonen el mismo por una operación de VCR. En el caso de que una petición al abandonar un multicast no tenga suficientes recursos para continuar con el servicio, sencillamente se cancelará el servicio y se generarán las correspondientes estadísticas.

Para poder simular la política de control de admisión se necesita mantener el estado (actual y futuro) de todos los recursos del sistema. El estado de los recursos es gestionado por las políticas de gestión de recursos, por lo cual el módulo de admisión de peticiones debe estar fuerte interrelacionado con estas políticas.

Los requisitos necesarios para poder atender una petición depende del tipo de política de servicio activa (en el caso de utilizar políticas multicast, la compartición de un canal de servicio existente puede reducir considerablemente los requisitos de la nueva petición), lo que implica una dependencia del sistema de admisión respecto a la política utilizada para servir una petición.

Políticas de gestión de recursos

Las políticas de gestión de recursos deben controlar los recursos disponibles (el ancho de banda de servicio, ancho de banda de red, almacenamiento, etc...) y los recursos ocupados en cada componente del sistema.

Los principales recursos que debe controlar el simulador son los anchos de banda de red y de servicio y el almacenamiento disponible en los servidores-proxy ó en los buffers de recepción de los clientes (lo cual posibilitará una futura implementación de políticas de servicio peer-to-peer ó chaining).

A la hora de controlar los recursos existen diversas posibilidades, en función del número de dimensiones que se tengan en cuenta a la hora de modelar el recurso en concreto.

Si consideramos el ancho de banda como un recurso unidimensional (figura 4-2a), entonces únicamente es necesario controlar el volumen de recursos libres y ocupados y la entidad del

sistema que los están utilizando. Por ejemplo, si el recurso que se está gestionando es un disco duro, utilizando el modelo unidimensional solo se controlaría el espacio del disco libre u ocupado del mismo y por lo tanto no se tendría en cuenta la posible fragmentación del disco.

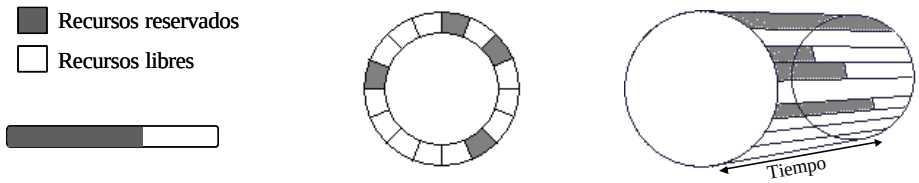
Si además del volumen de recursos reservados, también consideramos la ranura especifica del recurso que están asignadas entonces nos encontramos con un modelo de gestión de recursos bidimensional (figura 4-2b). Utilizando el ejemplo anterior del disco, estaríamos teniendo en cuenta qué porción del disco está reservada y cúal está libre. Este modelo es interesante siempre que la fragmentación del recurso pueda afectar a los resultados finales.

Los dos modelos anteriores suponen que los recursos siempre se reservan por un espacio de tiempo infinito desde el instante de tiempo actual. Sin embargo, la reserva de recursos se suele realizar teniendo en cuenta un horizonte temporal (que normalmente es el tiempo de duración de la petición que reserva los recursos). En el caso que los recursos se pueda reservar durante un tiempo finito (como ocurre con los ancho de banda de servicio y red) ó bien que se puedan planificar reserva futuras de los recursos, entonces se requiere una tercera dimensión para representar el estado del recurso en función del tiempo (figura 4-2c).

Los modelos bidimensional y tridimensional se ajustan mejor a las políticas de planificación rotativas (dentro de un determinado periodo de servicio) utilizadas por los servidores para asegurar un suministro continuo a los usuarios y una determinada calidad de servicio. En este caso la segunda dimensión consiste en cada una de las ranuras en que se puede dividir el periodo de servicio.

Los recursos de almacenamiento en los servidores-proxy se ajustan mejor al esquema bidimensional, de forma que se tenga en cuenta el fragmento de disco que se está reservando. En principio estos recursos se suelen reservar por periodos indefinidos de tiempos y por lo tanto no es necesario utilizar el modelo tridimensional.

El simulador soporta simultáneamente los tres modelos de gestión para los recursos, dejando en manos del usuario la configuración de cual de los tres tipos es el más idóneo para la experimentación que se quiere realizar. Por defecto se supone un modelo tridimensional tanto



(a) Modelo unidimensional (b) Modelo bidimensional (c) Modelo tridimensional

Figura 4-2. Modelos de gestión de los recursos soportados por el simulador

para el ancho de banda de servicio y de red y un modelo bidimensional para la gestión de los distintos tipos de almacenamiento.

Políticas de servicio

Las políticas de servicio son las encargadas de atender las peticiones de los usuarios y de decidir la mejor forma de servirlas con los recursos disponibles en cada momento.

Existen tres tipos de políticas de servicio diferentes para servir las peticiones en un sistema VoD, unicast, broadcast y multicast. Nosotros solo vamos a tener en cuenta las políticas de unicast y multicast, dejando las políticas de broadcast (ya que no permiten ofrecer servicios de TVoD) para una futura ampliación de la funcionalidad del simulador.

El servidor debe disponer de una política de servicio genérica que le permita decidir de que forma se atienden las distintas peticiones de los usuarios. Esta política debe decidir si se utilizan políticas como unicast, multicast, broacast, ó bien si se utilizan simultáneamente una combinación de éstas. En este último caso, se debe decidir cuándo se realiza una ú otra política y si esta decisión se toma estáticamente ó dinámicamente en función de la frecuencia de acceso de los contenidos multimedia.

Para poder simular el funcionamiento de estas políticas se tienen que tener en cuenta los recursos requeridos para servir la petición. Esto implica conocer el camino que va ha seguir el stream de servicio (las redes que se cruzarán) y el ancho de banda requerido para la transmisión del contenido seleccionado. En el caso de utilizar políticas de servicio multicast, también se debe llevar un control de las peticiones activas de forma que se pueda verificar si existe alguna petición con la cual se pueda compartir total ó parcialmente los recursos requeridos por la nueva petición.

El control de las peticiones activas se realiza de forma individual en cada uno de los servidores del sistema. Para ello se utiliza el concepto de canal de servicio, que especifica un flujo de información entre un fuente y uno ó varios destinos. Los canales de servicio identifican cada una de las transmisiones activas en el sistema. Desde el punto de vista del receptor el tipo del canal puede ser unicast (existe un único receptor), multicast (varios receptores) ó bien broadcast (0, 1 ó varios receptores).

Entre otra información, el canal de servicio debe especificar:

- La información que se va a transmitir por él, especificando el identificador y el fragmento del video (ya que se debe poder transmitir el video completo ó solo una porción del mismo) que se va a transmitir especificando el offset inicial y el final.

- El identificador del emisor y de los destinatarios de la información transmitida a través del canal. Normalmente, el emisor será un único servidor y los receptores pueden ser uno ó varios de los usuarios, ó cualquier servidor-proxy que quiera guardar el contenido multimedia en su cache.

- Temporización de la transmisión, es decir, cuándo se produce la transmisión y la recepción de la información transmitida. Esta información se utilizará tanto en el servidor (para implementar las políticas de planificación de los canales de servicio) como en los usuarios (para controlar la recepción de la información, actualización del estado de los buffers y la reproducción de los contenidos).

- Recursos asociados al canal de comunicaciones. El canal de servicio debe llevar el control de los recursos reservados tanto en el servidor (ancho de banda de servicio) como en la red de comunicaciones (ancho de banda de red).

La política de servicio primero tiene que comprobar si la petición del cliente se puede servir total ó parcialmente por un canal activo. En el caso de que no sea así, se debe crear un canal nuevo para que el cliente pueda recibir el video. La creación del canal (y con ello el servicio de la petición del usuario) se puede retardar con el objetivo de esperar la recepción de alguna nueva petición para los mismos contenidos que permita compartir los recursos del canal de servicio (política de servicio Batching). Cuando se permite retardar el servicio de las peticiones, se debe tener en cuenta la posibilidad de que el usuario decida no esperar más y finalmente opte por cancelar su petición.

Dentro del amplio rango de políticas de servicio multicast, el simulador soporta las política de batching y patching. Se han seleccionado estas dos políticas ya que consideramos que son las más representativas de las propuestas en la literatura.

La política de patching permite que un usuario se conecte a un canal de servicio ya iniciado (del cual obtendrá la mayor parte del stream de video) y al mismo tiempo se reciba el resto del video que falta (denominado prefijo) mediante un canal unicast propio. Para poder simular esta política se tiene que crear un canal de servicio adicional (canal de patching) por el cual se enviará la porción del contenido que falta, por lo tanto, los canales tienen que admitir que se transmita por ellos el video completo ó únicamente una fracción.

La política de patching, al contrario que ocurre con la de batching, permite ofrecer servicios TVoD y soporta los comandos VCR.

Políticas de gestión de contenidos

El simulador debe tener en cuenta tanto la ubicación distribuida de los contenidos así como los distintos esquemas de gestión de los mismos almacenados en los servidores-proxy.

Como ya hemos comentado, el simulador asume que los servidores estándar (utilizados por las arquitecturas centralizadas, servidores independientes y en el primer nivel de los sistemas basados en proxy), disponen de todos los videos que conforman el catálogo del sistema. Sin embargo, los servidores-proxy disponen de una capacidad de almacenamiento limitada que solo les permite almacenar un subconjunto de los videos del sistema. El simulador debe ser capaz de soportar la gestión de estos contenidos en los servidores-proxy.

En una primera instancia las políticas de gestión de contenidos que debe soportar el simulador son las políticas de caching y mirroring, ya que son las utilizadas por las arquitecturas de VoD estudiadas.

- Mirroring

  Para simular la política de mirroring, únicamente se requiere disponer de información sobre la capacidad de almacenamiento dedicada a este esquema y decidir la distribución de los contenidos multimedia del catálogo entre los distintos mirrors parciales del sistema simulado.

  La distribución de los contenidos se realiza utilizando una política de asignación de videos (estas políticas se describen más adelante en el capítulo 5 de la memoria). La asignación de los videos se realizará de forma estática durante la etapa de inicialización del sistema, una vez decidida la capacidad de almacenamiento dedicada a cada uno de los esquemas de gestión.

  Esta distribución inicial no se modificará durante la simulación, ya que se supone que el número de contenidos del sistema es constante (ya que la capacidad de los servidores también lo es) y que la única modificación en los contenidos que se realizan consisten en la sustitución de videos antiguos por otros nuevos.

- Caching

  La simulación de las políticas de caching resulta algo más complicada ya que requiere de la simulación de las políticas de selección de los contenidos que se deben almacenar en la cache, políticas de reemplazo de contenidos y la gestión de las estadísticas de acceso requeridas paras tomar las distintas decisiones.

  Existen distintos esquemas de caching que se pueden simular: caching autónomo, caching cooperativo y caching jerárquico. Inicialmente el simulador únicamente soportará el caching

autónomo y el caching cooperativo. La principales diferencias entre ambos esquemas estriba entre el rango de estadísticas que se tienen en cuenta a la hora de tomar las decisiones. En el caching autónomo sólo se tiene en cuenta información local, mientas que en el caching cooperativo se utilizan estadísticas conjuntas de los distintos vecinos.

Respecto al tamaño de los objetos almacenados en la cache, existen dos opciones: suponer que la cache almacena exclusivamente objetos multimedia completos ó bien que también se permite almacenar fragmentos de los objetos multimedia. De momento el simulador solo tiene implementadas políticas de caching de objetos multimedia completos, aunque en el diseño se ha tenido en cuenta futuras ampliaciones que permitan soportar políticas basadas en el almacenamiento en la cache de contenidos parciales, utilizando políticas de caching de prefijos ó segmentos.

Para soportar estas políticas se necesita que el simulador admita una gestión del almacenamiento a más bajo nivel, teniendo en cuenta la distribución de los distintos objetos dentro del disco, controlando si hay suficiente espacio libre para almacenar un determinado contenido y teniendo en cuenta la fragmentación externa que puede aparecer en el disco.

Por último, se necesita modelar las distintas políticas de reemplazo de contenidos de la cache. La política de reemplazo es la que decide que información se extrae de la cache cuando se necesita espacio para colocar otro contenido multimedia. Esta decisión se puede tomar de forma centralizada ó distribuida en función de si se utilizan estadísticas de acceso a los contenidos locales ó globales. También se debe especificar cuándo y cómo se generan las estadísticas de acceso a los contenidos y durante cuánto tiempo los valores son validos (envejecimiento de las estadísticas).

La simulación de la política de gestión de los contenidos de la cache consta de distintas etapas:

1. Cálculo ó actualización de las estadísticas de los contenidos. Las estadísticas de acceso a los contenidos multimedia (en el caso de que se tenga en cuenta estadísticas conjuntas de los distintos servidores) no se pueden actualizar cada vez que se acceda a un contenido susceptible de ser almacenado en la cache, ya que esto podría sobrecargar los servidores ó saturar las redes de comunicación. Para evitar este efecto las estadísticas solo se actualizar cada cierto tiempo.
2. Decidir si el contenido accedido es susceptible de ser almacenado en la cache del servidor local.

3. En el caso de que no haya suficiente capacidad de almacenamiento libre, elegir el contenido ó los contenidos que deben ser expulsados para hacer hueco al nuevo contenido. Esta decisión se tomará en función de la política de reemplazo y teniendo en cuenta la frecuencia de acceso de cada uno de los contenidos de la cache.

El simulador deberá realizar una eliminación progresiva de los contenidos de la cache que están siendo utilizados actualmente para servir una petición. En este caso se asume que la expulsión de la cache se realiza gradualmente a medida que se recibe el nuevo contenido, proporcionando un margen de seguridad para que la petición en curso pueda seguir sirviéndose en su totalidad desde la cache.

4.3 Modelo del simulador

Las simuladores se pueden clasificar en función de como avanza el tiempo durante la simulación. Se puede realizar una primera clasificación (en función del comportamiento del modelo a simular) entre sistemas continuos o discretos. En los sistemas continuos el tiempo fluye suavemente y constantemente. Mientras que en los sistemas discretos el sistema que está siendo simulado sólo cambia de estado en puntos discretos del tiempo de simulación. La simulación salta de un estado a otro cada vez que se sucede un evento.

Los simuladores también se pueden clasificar en función de si la progresión de la simulación avanza a saltos constantes (simulación dirigida por tiempo) o no (simulación dirigida por eventos). En los simuladores dirigidos por eventos (hechos significativo ocurridos que varían el estado del sistema) la variación del tiempo de simulación dependerá de cuando se produzcan los eventos y por regla general suele ser muy variable.

En el diseño del simulador de arquitecturas de VoD hemos optado por un simulador dirigido por eventos discretos. Esta opción se ha escogido debido a la propia naturaleza del sistema a simular y para lograr un mayor la velocidad de simulación.

Para crear el modelo de un sistema de VoD (centralizado ó distribuido), el número de aspectos que pueden ser potencialmente considerados son múltiples y variados. Para reducir la complejidad del modelo y el software desarrollado vamos a restringir éstos a los estrictamente necesarios para poder evaluar los aspectos de rendimiento y escalabilidad de las distintas arquitecturas. Aquellos aspectos que no influyan directamente en los objetivos de nuestra investigación se abstraerán y se simplificarán (siempre que sea posible) en la especificación del modelo del sistema.

Los principales características del modelo utilizado se comentarán a continuación: las simplificaciones estructurales supuestas en el modelo, el modelado de la carga del sistema, la descripción de los eventos definidos y por último, el modelado del ciclo de vida de las peticiones en el sistema.

4.3.1 Simplificaciones estructurales

El modelo de un sistema puede llegar a ser tan detallado que apenas se diferencie del sistema físico simulador. Sin embargo, a medida que se incrementa el nivel de detalle del modelo el simulador requerido es más complejo y se reduce la velocidad de simulación.

Es importante ajustar el nivel de detalle del simulador de forma que se reduzca su complejidad, pero manteniendo al mismo tiempo la validez del modelo. Esto se puede lograr abstrayendo parcialmente aquellos módulos que no estén estrechamente relacionados con los parámetros estudiados del sistema físico.

En el caso de nuestro modelo de simulador, denominando VoDSim, hemos tratado de simplificar algunas de las componentes / políticas que no afecten a la escalabilidad y rendimiento de las arquitecturas VoD estudiadas. Estas simplificaciones se han efectuado manteniendo la funcionalidad de los distintos elementos pero simulándolas con un menor nivel de detalle ó asumiendo ciertas premisas que faciliten su simulación.

A continuación, se describirán los distintos aspectos del modelo que se han simplificado para facilitar su implementación.

Modelo Routing

Para simplificar el modelo del encaminamiento (rounting) en las redes del sistema vamos asumir la existencia de un único camino (el más corto) entre cada par de componentes del sistema. De esta forma, siempre que se deba tomar una decisión de encaminamiento se escogerá el camino que minimiza el número de redes que se deben cruzar para alcanzar el destino. Esta simplificación no contempla las posibles políticas de balanceo de tráfico que pudiesen implementar internamente los componentes de comunicación (routers).

Capacidad de servicio

Todos aquellos aspectos del servidor de VoD que no afecten directamente a nuestra investigación han sido modelados a alto nivel para reducir en lo posible la complejidad del simulador.

En concreto, todos los aspectos concernientes a la generación de los streams de servicio (ancho de banda de disco, gestión de los buffers de memoria, planificadores de disco, etc...) que caracterizan la capacidad de servicio del servidor, han sido sustituidos por un único parámetro: el ancho de banda de servicio. Este ancho de banda de servicio especifica la capacidad de gestión de streams independientes del servidor. Este parámetro en conjunción con la velocidad de visualización *(play rate*) de los videos del sistema, definirán el número máximo de usuarios independientes (sin compartir recursos) que es capaz de soportar un servidor.

Modelo de la red de transmisión en tiempo real

Al igual que con los servidores, para modelar la capacidad de transmisión de las redes del sistema también se han utilizado anchos de banda de transmisión efectivos. Estos anchos de banda tratan de abstraer la complejidad de la gestión interna de los componentes y protocolos de red (colisiones, perdidas de paquetes, estructura de datos de los protocolos, etc...) que reducen la eficiencia del ancho de banda físico de estos componentes.

Además, también se supone que la infraestructura de red tiene incorporados los mecanismo necesarios (mediante la utilización de redes ATM ó protocolos de tiempo real RTP, RSVP, etc...) para reservar el ancho de banda y garantizar la transmisión en tiempo real (con una latencia acotada) que asegure una calidad de servicio. De acuerdo a estas suposiciones, el simulador asume tiempos de llegada de los datos dentro de los intervalos de tiempos requeridos para una correcta reproducción de los contenidos multimedia.

Caracterización de los contenidos multimedia

El modelo utilizado para los contenidos supone una velocidad de reproducción de los mismos constante a lo largo de todo el video (CBR). Esta característica se puede lograr en los sistemas reales, incluso utilizando contenidos VBR, mediante la utilización de diversas políticas de *smoothing* ó *prefetching* que permiten a los servidores gestionar los contenidos multimedia como si fuesen verdaderos contenidos CBR.

4.3.2 Modelo de la carga de trabajo del sistema

A la hora de realizar un análisis de rendimiento basado en un modelo analítico ó bien si se utiliza la simulación, es imprescindible la especificación de un modelo parámetrizado de la carga de trabajo que soporta un sistema físico.

La caracterización de la carga de trabajo implica analizar la naturaleza de los servidores, redes y usuarios, sus características y el consiguiente desarrollo de un modelo flexible que

permita analizar el sistema bajo distintas premisas. Una vez se ha desarrollado el modelo de carga, se puede evaluar el efecto de los cambios en el sistema y la carga mediante la modificación de los parámetros del modelo.

Una alternativa a la utilización de un modelo de carga sintético es la utilización de medidas reales de la carga de trabajo obtenidas a partir de un fichero de traza de un sistema real. Sin embargo, lamentablemente en el ámbito de los sistemas de VoD estas trazas todavía no están disponibles.

Los principales parámetros que intervienen en el modelo de trabajo soportado por un sistema VoD están relacionados con el comportamiento del usuario y en menor medida con ciertas características del sistema.

Desde el punto de vista del diseño del simulador se ha intentado no fijar un modelo de carga especifico, para desligar el modelo de comportamiento de los usuarios y del sistema de una implementación específica e inamovible del simulador. Este objetivo se ha logrado definiendo y parámetrizando los principales componentes que caracterizan la carga, pero dejando en manos del usuario final la especificación del modelo concreto que debe seguir cada uno de ellos.

Nuestra modelo de carga del simulador se basa en los siguientes parámetros:

- Velocidad de llegada de peticiones de los usuarios (Request Rate).

  Este parámetro se utiliza para especificar el volumen de peticiones que debe soportar el sistema, especificando el intervalo de tiempo que transcurre entre dos peticiones consecutivas. Existen diversas formas de modelar la velocidad de llegada de peticiones al sistema [Gri00], aunque la más utilizada se basa en la utilización de un proceso de Poisson con media de $1/\lambda$, donde λ es la velocidad de llegada de peticiones al sistema [Dan94].

  Según esta definición, el volumen de peticiones que tiene que soportar el sistema es independiente de su tamaño. Para corregir este defecto hemos optado por permitir la redefinición de la velocidad de llegada de forma que tenga en cuenta el tamaño del sistema (en función del número de redes locales ó del número de usuarios conectados). De esta forma, permitimos la definición de la velocidad de llegada de las peticiones de forma local a cada una de los servidores del sistema.

- Tipos de peticiones.

  Existen dos aproximaciones para modelar la interrelación entre los usuarios y el sistema de VoD. En la primera de ellas, se asume un modelo de iteración en el cual los usuarios únicamente pueden acceder a un contenido y reproducirlo, no permitiendo la realización de

ningún otro tipo de comando. Este es el modelo básico asumido para la versión actual del simulador.

La segunda aproximación tienen en cuenta, que el usuario también pueda realizar comandos de VCR (pausa, avance y retraso rápido, búsqueda, etc...). Este nivel de iteración requiere modelar los hábitos de los usuarios a la hora de utilizar estos comandos [Bra99].

- Tiempo de espera máximo.

  Este parámetro interviene cuando se utilizan servicios Near-VoD que no garantizan la atención inmediata de las peticiones de los usuarios. En este caso, es necesario modelar la posibilidad de que el usuario decida no esperar más tiempo y cancele su petición.

  El modelado del tiempo de espera se suele basar en un tiempo medio de espera más una cierta cantidad de tiempo variable que se calcula en función de una determinada distribución (la exponencial por ejemplo).

- Popularidad de los contenidos.

  La popularidad de los contenidos es uno de los parámetros más relevantes para el rendimiento del sistema cuando se utilizan políticas multicast ó esquemas de caching.

  A la hora de modelar la popularidad de los contenidos de un sistema de VoD, la distribución zipf [Zip49] ha demostrado ofrecer una buena aproximación [Dan94]. Esta aproximación proporciona un modelo estático que si bien funciona correctamente en periodos cortos de tiempo, no es adecuada para desarrollo a largo plazo.

  Para compensar esta carencia, Dan y Sitaram [Dan93] han propuesto un modelo basado en una rotación de las probabilidades de los contenidos. Recientes trabajos [Gri97] han propuesto un modelo de popularidad de los contenidos que permita separar el ciclo de vida de los videos (entendido como la variación de su popularidad a medida que transcurre el tiempo) del comportamiento del usuario.

  La versión actual del simulador soporta básicamente un modelo estático para la popularidad de los contenidos.

- Tamaño de los contenidos.

  El simulador soporta la parametrización del tamaño de los contenidos. Se puede especificar que todos los contenidos tengan un mismo tamaño ó bien que éste dependa de cada contenido (seleccionado el tamaño de forma aleatoria en función de una determinada distribución).

- Distribución de las peticiones entre los usuarios.
  Para identificar cual de los clientes ha realizado una petición, se utiliza una distribución uniforme entre todos los usuarios inactivos. Preferimos la utilización de distribuciones uniformes para modelar este parámetros ya que asumimos que todos los usuarios del sistema VoD tienen la misma probabilidad de realizar una petición.

- Velocidad de visualización (Play rate).
  El ancho de banda requerido para servir una petición dentro del sistema va a depender entre otros parámetros, del formato de video utilizado, el ratio de compresión y la resolución de la imagen.

  El simulador permite modelar los distintos formatos de video mediante la especificación de la velocidad de visualización del video. Recordemos que en la tabla 1-1 del apartado 1.1.1 mostramos la correspondencia entre los distintos formatos de video y sus respectivas velocidades de visualización.

  Por otro lado, como ya hemos comentado, suponemos que el ratio de compresión (y por lo tanto la velocidad de reproducción) de los contenidos es constante a lo largo de todo el video (CBR) en contraposición del modelo VBR que supone que el porcentaje de compresión puede ser diferente para cada fragmento del contenido. De esta forma, logramos simplificar el diseño e implementación del simulador, ya que no es necesario tener en cuenta políticas de suavización (smoothing [Fen95]) de los contenidos para evitar la perdida de la secuencia de visualización.

Los parámetros comentados a continuación permiten adaptar el modelo de carga del sistema (dentro de las limitaciones ya comentadas) a las propias necesidades de cada experimentación.

4.3.3 Eventos

Dentro del modelo de simulación por eventos, éstos representan las transiciones en el estado del sistema simulado. Los eventos representan la dinámica del sistema simulado, en contraposición del estado de simulación que aporta una visión estática del sistema.

En los sistemas de VoD este comportamiento dinámico está originado en el comportamiento de los usuarios del sistema y por el servicio de las peticiones generadas por estos últimos.

A continuación comentaremos la funcionalidad y las principales características de cada uno de los eventos utilizados en la herramienta de simulación.

Petición de usuario (VoDUserRequestEvent)

Este evento representa una petición realizada por parte de un usuario. Esta petición puede verse como un comando que se genera desde el cliente hasta el servidor.

Los parámetros requeridos para modelar una petición en el sistema son: el identificador del usuario, el tipo de comando que se realiza, el servidor al cual va dirigido inicialmente la petición y toda aquella información dependiente del tipo de petición solicitada.

Inicialmente el simulador sólo soporta los comandos de reproducción y parada. Estos comandos son más que suficientes para analizar la escalabilidad y la capacidad de servicio de las distintas arquitecturas LVoD. De todas formas, el diseño del simulador (aunque aún no sean necesarios) ha tenido en cuenta la futura introducción de los comandos VCR (pause, goto, fast-rewind y fast-forward) y se ha creado el soporte software necesario para su incorporación.

Cancelación de una petición (VoDRequestRenegeEvent)

Este evento modela la cancelación de una petición por parte de un usuario debido a que el sistema no le ha podido atenderla en un tiempo prudencial (tiempo de denegación del usuario).

Este evento solo se genera cuando se utilizan servicios Near-VoD ó políticas de compartición de recursos del tipo Batching que no aseguran el servicio inmediato de las peticiones de los usuarios. También se puede producir una cancelación de una petición debido a que el sistema no dispone de suficientes recursos para atenderla inmediatamente y, por lo tanto, decide retardar el servicio de la misma.

El principal parámetro que afecta a este evento es el tiempo que esta dispuesto a esperar el usuario antes de cancelar la petición, el cual actualmente está fijado a 10 minutos.

Tiempo expirado (VoDTimeExpiredEvent)

Este evento se utiliza para implementar un sistema de temporización requerido por las distintas funcionalidades dentro del simulador.

Entre otras cosas, este evento se utiliza para identificar el tiempo máximo que está dispuesto a esperar un usuario antes de cancelar su petición ó bien para la planificación de peticiones cuando se utilizan políticas de servicio *Batching*.

Rechazo de petición (VoDRefusedRequestEvent)

Con este evento se notifica que el sistema ha rechazado el servicio de una petición. Esta condición se puede producir por dos razones: como respuesta del servidor a la retirada de una

petición por parte del usuario ó bien cuando el sistema no dispone de suficiente recursos para servir una petición y se está utilizando una política de servicio de VoD verdadero.

Reconocimiento de la petición (VoDAckResponseEvent)

Este evento se genera cuando el sistema admite una petición realizada por un usuario. Este evento nos permitirá actualizar el estado del usuario en función del comando realizado y sobre todo, para calcular estadísticas de tiempo de respuesta del sistema (tiempo transcurrido entre que el usuario realiza la petición y ésta empieza a servirse).

Dentro de este evento se envía toda la información necesaria para que el usuario pueda empezar la recepción de los datos (identificador de los canales de recepción involucrados, etc...).

Inicio de envío de datos (VoDDataSendEvent)

Este evento permite modelar la funcionalidad del planificador de servicio dentro de los servidores de video. En concreto, el evento representa la transmisión por parte del servidor de una porción de video hacia un componente del sistema. La transmisión se realiza por el canal de comunicaciones asociado con la petición que se está sirviendo.

La información asociada con el evento debe especificar el origen y el destino de la transmisión, el identificador del canal utilizado y los datos que se están enviando.

Si la transmisión del contenido a través del canal de datos asociado con este evento no ha finalizado (se comprueba el offset del stream enviado con respecto al tamaño de la transmisión), se tiene que planificar el siguiente evento de envío de datos para el siguiente ciclo del planificador de servicio de red del servidor. Esta realimentación permite generar todas las transmisiones necesarias para completar una petición.

Final de envío de datos (VoDEndDataSendEvent)

El evento de final de envío modela la llegada de datos a los usuarios asociados con una canal de transmisión.

La razón por la cual se ha decidido utilizar 2 eventos distintos para modelar las transmisiones de la información entre los componentes del sistema estriba en que un canal de transmisión puede estar compartido por diferentes usuarios y, por lo tanto, tener varios destinos. Cada uno de estos destinos se pueden localizar en diferentes redes y, por lo tanto, requerir tiempos de llegada de la información distintos. Por esta causa, se necesita un modelo para las comunicaciones que nos permita desligar el instante de tiempo de inicio de la transmisión del tiempo de llegada a cada

uno de los destinos. Esto finalmente se consigue utilizando eventos dferentes para el envío y la recepción.

El cálculo de la duración de transmisión de los datos se calcula teniendo en cuenta el retardo (latencia) introducido por las redes que tienen que cruzar los datos antes de llegar al usuario. El instante concreto de la llegada de un paquete de datos se registra en el instante en que llega el último byte de información al destino.

Final de petición (VoDUserRequestEvent)

El simulador requiere conocer en todo momento cuando se finalizan las peticiones de los usuarios de forma que se pueda llevar un control efectivo de los recursos disponibles en cada instante de tiempo.

El objetivo del evento de finalización de una petición es permitir al motor de simulación liberar todos los aquellos recursos reservados y actualizar el estado de los usuarios.

4.3.4 Ciclo de vida de una petición

En la figura 4-3 mostramos el ciclo de vida que sigue una petición una vez entra en el sistema, así como los distintos módulos y/ó políticas que intervienen en cada una de las etapas.

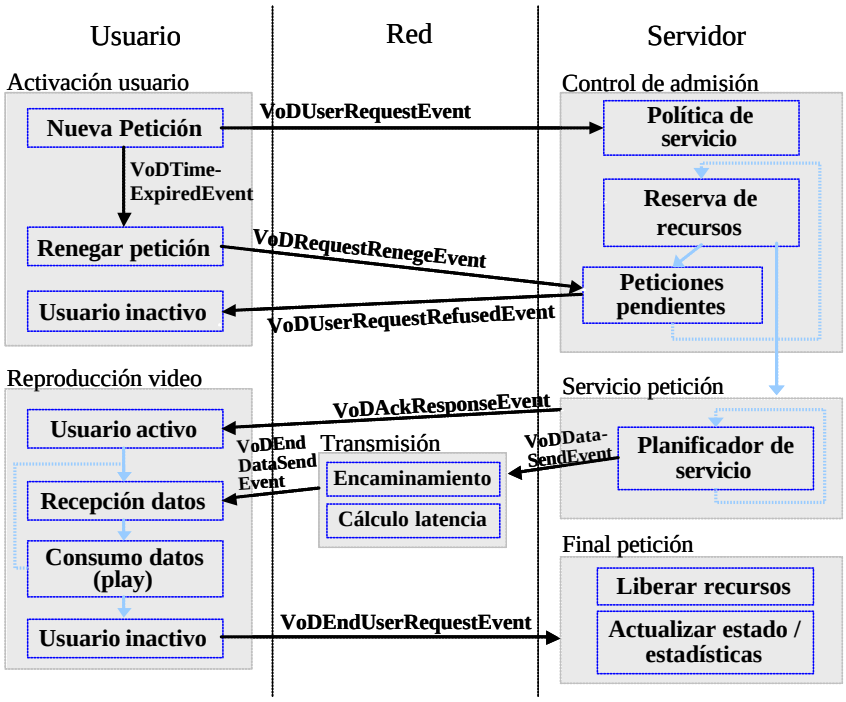


Figura 4-3. Ciclo de vida de una petición en el simulador.

Las peticiones se arrancan cuando un usuario ejecuta un comando de visualización de un video, realizando una solicitud al servidor que tiene asignado mediante el evento *VoDUserRequestEvent*.

Al mismo tiempo que se realiza una petición al servidor, se planifica un evento de expiración de tiempo para modelar el tiempo máximo que el usuario está dispuesto a esperar el inicio del servicio.

Cuando el servidor correspondiente recibe la petición, la pasa al módulo de control de admisión. Este módulo, en función de la política de servicio utilizada y la posibilidad de poder compartir un canal de transmisión con otros usuarios, evalúa los recursos necesarios (contenido multimedia, ancho de banda de servicio y ancho de banda de red). En el caso de que todos los recursos estén disponibles se produce su reserva y se inicia el servicio de la petición.

En el caso de no existir el contenido solicitado en el servidor, la petición se redirige hacia el siguiente nivel en la jerarquía de servicio de la arquitectura utilizada. Si este nivel finalmente no existe, entonces se rechaza la petición y se informa al usuario mediante el evento *VoDRequestRefusedEvent*. Si el problema consiste en la carencia de cualquiera de los otros recursos, entonces dependiendo del tipo de política de servicio, la petición se almacena en una cola de peticiones pendientes para un servicio posterior ó directamente se rechaza.

Las peticiones pendientes salen de la cola cuando existen suficientes recursos para poder servirlas ó bien cuando el usuario cancela su petición porque se ha superado el tiempo máximo que estaba dispuesto a esperar. En este último caso se genera el evento de *VoDRequestRenegeEvent*, al cual el servidor contestará confirmando el rechazo de la petición, con la consiguiente desactivación del usuario y la actualización de las estadísticas del sistema.

Cuando una petición ya ha sido admitida por la política de control de admisión se inicia el servicio de la petición por parte del planificador de servicio, y se envía al usuario de la confirmación de la aceptación (*VoDAckRequestEvent*). El planificador se encarga de crear ó actualizar el canal de transmisión asignado a la petición, y a partir de ese instante, transmitir en cada turno de servicio la porción del video seleccionado.

Una vez transcurrido el tiempo necesario para la transmisión de la información, ésta es recibida por el usuario correspondiente. La recepción de cada fragmento de video por parte de los usuarios implica la actualización del estado de su buffer de recepción y la actualización del puntero de visualización que permitirá controlar la finalización de la reproducción del video por parte del usuario.

Una vez el puntero de visualización alcanza el final del video, el usuario informa de la finalización de la petición al servidor (mediante el evento VoDEndUserRequestEvent) e

inmediatamente se desactiva el usuario. La finalización de una petición implica desde el punto de vista del servidor, la liberación de los recursos y la actualización de las estadísticas del sistema.

4.4 Configuración de los parámetros de simulación

El diseño y el modelo presentado en el apartado anterior se han reflejado en la implementación de la herramienta del simulador (presentada en el apéndice A) y en la especificación de los diversos parámetros que nos permitirán caracterizar el comportamiento de los sistemas LVoD analizados.

Una de las características de los sistemas VoD es la gran cantidad de componentes y variables que intervienen en el sistema. Esto conlleva que su simulación requiera de un gran número de parámetros y que si no se diseña con cuidado su especificación puede resultar sumamente tediosa y complicada. A la hora de acometer este problema hemos tenido en cuenta la futura portabilidad de la herramienta de simulación a otros entornos (Internet entre ellos) y la necesidad de un método que nos permitiese incorporar nuevos parámetros al simulador de forma flexible y sin requerir grandes cambios.

Después de sopesar distintas alternativas nos hemos decidido por la utilización del estándar XML para definir los parámetros de la simulación. Esta configuración se basa en la definición de dos archivos: un archivo de definición del conjunto parámetros que se pueden configurar y un fichero de configuración de la simulación que especifica los valores asignados a los parámetros de simulación.

El fichero de definición de parámetros (*VoDsimulator.dtd*) tiene por objetivo especificar la sintaxis y los valores por defecto (en el caso de que su especificación no sea obligatoria) de los parámetros de configuración del sistema VoD a simular. Este fichero se basa en el formato DTD (Data Type Definition) que nos permite verificar si un determinado documento XML es válido. Mediante la definición de tipo de documento (DTD) se describe el contenido y la estructura de un determinado tipo de documento XML (en nuestro caso el fichero de configuración de la simulación), especificando las marcas XML válidas, el orden de las marcas, si las marcas son opcionales, el tipo de datos contenido en las marcas, los atributos de las marcas, los valores de aquellos atributos que tienen un valor fijo, las instrucciones de proceso, las entidades, etc... En el apéndice A (apartado 2) mostramos la definición completa de la DTD que especifica los parámetros de configuración del simulador.

El fichero de configuración por su parte, define los valores asignados específicamente a un conjunto de parámetros de la simulación. A la hora de configurar una determinada simulación,

solamente se necesita especificar aquellos parámetros cuyos valores asignados son diferentes de los asumidos por defecto.

Los parámetros de configuración del simulador se pueden agrupar básicamente en 3 categorías:

- Configuración del experimento ó prueba.

  Dentro de esta categoría entran todos aquellos parámetros que permiten especificar el tipo de experimento que se quiere realizar con el simulador y los ficheros en donde se generan los resultados de la simulación.

  Algunos de los parámetros que forman parte de la configuración del experimento son: el número de simulaciones que se realizan de cada experimento, los ficheros de estadísticas y traza utilizados, el tipo de estudio que se quiere realizar (escalabilidad, influencia del ratio de peticiones, influencia de la distribución de almacenamiento entre los distintos esquemas de gestión, influencia de la capacidad de almacenamiento de los servidores-proxy, etc...) y por último, las condiciones de finalización de la simulación.

  En la tabla 4-1 mostramos los principales parámetros de configuración de la simulación ó el tipo de prueba.

Tabla 4-1. Parámetros de configuración del experimento

Parámetro	Explicación	Valor	Por defecto
TestName	Nombre simulación	cadena	Ninguno
StatisticsFile	Nombre fichero estadísticas	Ruta fichero	Ninguno
OutputFile	Nombre fichero traza salida	Ruta fichero	Ninguno
PrintEvents	Eventos generados en la traza	All, MainEvents, None	None
WriteOutput	Indica si se genera la traza	(Yes, No)	No
RandomEvents	Generación aleatoria eventos	(Yes, No)	No
SymTest	Tipo de prueba, para cada una de las pruebas podemos especificar parámetros adicionales	Simple, Scalability, Mean, Storage, StorageDistribution, OptimalMeanMoviesDistance, RequestRate, MeanDistanceCorrelation	Simple

- Configuración del sistema de VoD simulado.

  Esta categoría agrupa todos aquellos parámetros que afectan a la arquitectura del sistema de VoD. Estos parámetros nos permiten especificar el tipo de arquitectura a simular, las políticas de gestión activas, el número de componentes del sistema (redes locales y servidores), así como la configuración especifica para cada uno de estos componentes (ancho de banda, número de puertos, capacidad de almacenamiento, etc...).

En la tabla 4-2 se describen los principales parámetros que permiten configurar el sistema LVoD a simular.

Tabla 4-2. Parámetros de configuración del sistema LVoD

Parámetro	Explicación	Valor	Por defecto
Type	Tipo de arquitectura	Centralized, IndependentServers, OneLevelProxy, PTree, DoublePTree	Ninguno
UseSegmentedNetworks	Si se utilizan redes segmentadas	Yes, No	No
NetworkPorts	Puertos redes	Número	12
ServerPorts	Puertos de servicio	Número	1
TrafficBalancing	Si esta activada la política de balanceo de peticiones	Yes, No	No
ServersNumber	Número de servidores	Número	Ninguno
LocalNetworks	Número de redes locales	Número	Ninguno
InitCache	¿Se inicializa la cache autónoma?	Yes, No	No
TreeOrder	Orden árbol	Número	2
WithMainServer	Si hay servidor principal	Yes, No	No
InitCollaborativeCache	¿Se inicializa la cache cooperativa?	Yes, No	No
BestCacheMirrorDistribution	Si se calcula la mejor distribución cache / mirror	Yes, No	No
StorageDistributionPolicy	Politica de distribución del almacenamiento	Default, Global, Adaptive	Default
MirrorDistributionPolicy	Forma de recorrer el árbol cuando se realiza el mapping	ByIndividualMovies, ByIndividualMoviesPostOrder, ByBalancedMoviesGroups, ByBalancedMoviesGroupsPostOrder	ByIndividualMovies
MirrorMappingPolicy	Política de asignación de videos	Secuencial, ByMoviesDistance	ByMoviesDistance
BuildingMirrorStepByStep	Si se realiza un mapping gradual	Yes, No	No
MoviesToMapEveryStep	Número de videos asignados en cada paso	Número	1
TakeInAccountCollateralEffects	Se tiene en cuenta los efectos colaterales	Yes, No	No
Bandwitdh	Ancho de banda	Número	Ninguno
ServicePolicy	Política de servicio	Unicast, Batching, Patching	Unicast
ResourceManager	Gestor de recursos	Normal, Slot, Complex	Slot
StorageCapacity	Capacidad de almacenamiento del servidor	Número	Ninguno
CachingPolicy	Política de gestión de la cache	FullMovieCaching, FullMovieCollaborativeCaching Policy, PrefixCaching	
MirrorCapacity	Capacidad del mirror	Número	Ninguno
PortsNumber	Número puertos red	Número	12
ReplacementPolicy	Política de reemplazo en la cache	NoReplacement, LessFrecuentAccessed , LessPopular	LessFrecuentAccessed

ReplacementUmbral	Umbral de reemplazo	Número	Ninguno
OnlyReplaceNotUsedMovies	Si sólo se reemplza los videos no utilizados	Yes, No	No
DoAgeing	¿Envejecimiento estadísticas?	Yes, No	No
AgeingFrecuency	Frecuencia de envejecimiento	Número	Ninguno
CollaborativeCachingCapacity	Capacidad de la cache cooperativa	Número	Ninguno
CachingMovieSelectionPolicy	Política de selección de los videos en la cache	MostAccesedLimited, ByMovieReplication	MostAccesedLimited
MoviesReplication	Factor de replicación en la cache	Número	2
MoviesCachingPriorityPolicy	Método de inicialización de los videos en la cache cooperativa	Default, None, ByAchieveReplicationPercentage, ByMovieDistance, ByMovieDistanceAndReplicationPercentage	Default

- Configuración del modelo de carga de trabajo del sistema y comportamiento de los usuarios.

  Esta categoría hace referencia a todos aquellos parámetros que influyen en el modelo de carga del sistema simulado ó al modelo del comportamiento de los usuarios. A través de estas opciones se pueden especificar el número de contenidos disponibles en el sistema y su tamaño, número de usuarios conectados al sistema y su distribución en el sistema, el modelado de la selección de los videos por parte de los usuarios, modelo de comportamiento de los usuarios, etc.

  La tabla 4-3 muestra los parámetros que permiten configurar el modelo de carga de la simulación.

Tabla 4-3. Parámetros de configuración del modelo de carga del simulador

Parámetro	Explicación	Valor	Por defecto
MoviesSize	Tamaño de los videos	Distribución	Uniform(90)
MoviesPlayRate	Velocidad de reproducción de los videos	Distribución	Uniform(1500 Kb/s)
MoviesNumber	Número de videos catálogo	Número	100
RequestInterval	Frecuencia de llegada de peticiones	Distribución	Poisson (10)
RequestMovie	Selección de videos por parte de los usuarios	Distribución	Zipf (0.728)
RenegingInterval	Tiempo de cancelación de las peticiones	Distribución	Uniforme(10)
UsersNumber	Número de usuarios del sistema	Número	Ninguno
UsersServerDistribution	Distribución de los usuarios entre los servidores	Uniform, NonUniform, OneLevelProxies	Uniform
BufferSize	Tamaño del buffer de usuario	Número	5 minutos

En el apéndice A (apartado 2) adjuntamos los ficheros de configuración ejemplo para cada una de las arquitecturas Proxy-Tree y Double P-Tree.

4.5 Validación de la herramienta de simulación VoDSim

En este apartado se procede a la validación de la herramienta de simulación. Debido a que no se dispone de un sistema real de LVoD con las características utilizadas en nuestro estudio (más 100 redes / servidores) se ha optado por validar los resultados de simulación contra los resultados obtenidos mediante los modelos analíticos presentados a lo largo de la presente memoria.

La verificación se realizará en función de las principales arquitecturas analizadas a lo largo de la memoria: servidores-proxy de un nivel, Proxy-Tree y Double P-Tree. Cuando las arquitecturas se basen en la utilización de redes no segmentadas, se validará el simulador contra los resultados inferidos a partir de la distancia media de servicio de la arquitectura obtenida mediante la expresión (3.8) del capítulo 3. Mediante la distancia de servicio (D_{ms}) el ancho de banda efectivo del sistema (B_e) se obtiene con:

$$B_e = \frac{N \cdot B_c}{D_{ms}} \qquad (4.1)$$

En este caso se puede aproximar el ancho de banda efectivo (B_e) dividiendo el ancho de banda de red (número de redes locales, N, por el ancho de banda de cada red local, B_c) entre la distancia media de servicio (D_{ms}).

4.5.1 Validación de la arquitectura de servidores-proxy de un nivel

Primero vamos a verificar los resultados proporcionados por el simulador para las arquitecturas de servidores-proxy de un nivel. Para realizar una validación que sea independiente del tamaño del sistema ó de los recursos del mismo vamos a comprobar los resultados para arquitecturas de distintos tamaños (desde 15 hasta 255 redes), utilizando redes no segmentadas y distintos capacidades de almacenamiento en los servidores-proxy.

La figura 4-4 muestra el ancho de banda efectivo (capacidad de servicio del sistema) obtenido por un sistema de servidores-proxy de un nivel a medida que se incrementa el número de redes locales del sistema. Se prueban dos configuraciones diferentes para los servidores-proxy: en la primera el proxy tiene suficiente capacidad para almacenar el 30% de los videos del sistema y en la segunda una capacidad para el 40%.

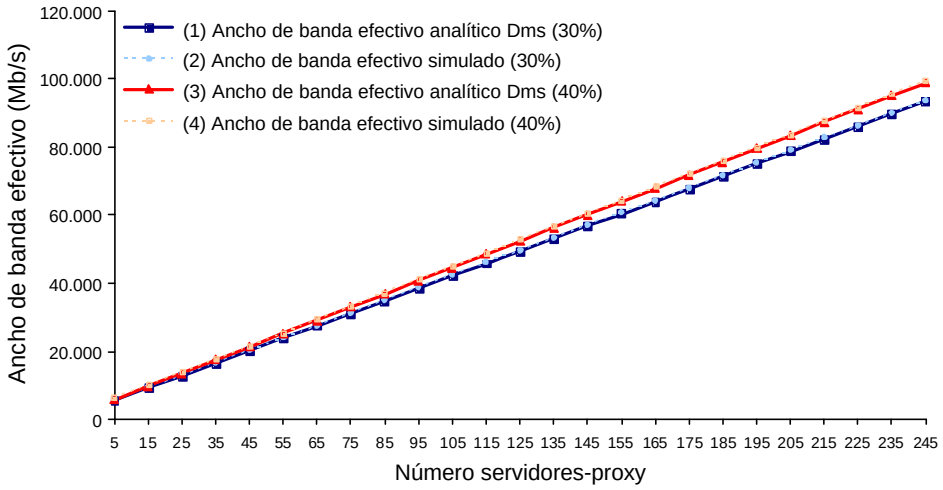


Figura 4-4. Validación de la arquitectura de un nivel de servidores-proxy utilizando redes no segmentadas

Las gráficas (1) y (3) muestran el ancho de banda efectivo anterior calculado a partir de la distancia media de servicio (utilizando la expresión 4.1) y suponiendo un ancho de banda para cada red local de 500 Mb/s, mientras que las gráficas (2) y (4) muestran los mismos parámetros de rendimiento pero esta vez calculados mediante la herramienta de simulación.

Si comparamos ambos resultados podemos comprobar que los resultados obtenidos mediante la simulación (gráficas 2 y 4) se superponen perfectamente a los obtenidos a partir de la distancia media de servicio (gráficas 1 y 3). Esta correlación nos permite comprobar la validez de la herramienta de simulación para las arquitectura de servidores-proxy de un nivel, independientemente de los recursos disponibles y el tamaño del sistema de LVoD.

4.5.2 Validación de la arquitectura Proxy-Tree

En la validación de la arquitectura Proxy-Tree vamos a seguir los mismos pasos que con la arquitectura anterior, contrastando los resultados de ancho de banda efectivo de la simulación con los obtenidos mediante el modelo analítico basado en la distancia de servicio.

La figura 4-5, muestra el ancho de banda efectivo para la arquitectura Proxy-Tree obtenido por la simulación (gráficas 3 y 6) y el ancho de banda efectivo inferido a partir de la distancia de servicio teórica calculada mediante la expresión 3.4 (gráficas 1 y 4) ó bien utilizando la distancia de servicio real (gráficas 2 y 5). La distancia media de servicio real se obtiene a partir del sistema simulado, una vez se ha realizado la distribución de los contenidos entre todos los servidores-proxy.

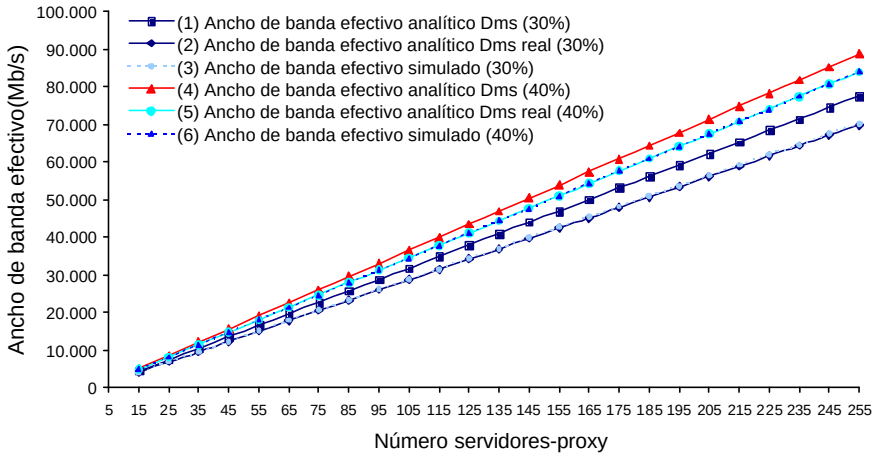


Figura 4-5. Validación de la arquitectura Proxy-Tree con redes no segmentadas

Si nos fijamos en los resultados obtenidos mediante el modelo analítico basado en la distancia de servicio teórica (gráficas 1 y 4), podemos observar que la diferencia con respecto a la simulación es de apenas un 5% cuando se utiliza una capacidad del 40% y del 9% cuando se utiliza una capacidad de almacenamiento del 30%, siempre a favor del modelo analítico.

La desviación entre los ambos resultados se justifica porque a la hora de calcular la distancia media de servicio teórica se asume una distribución óptima de los contenidos entre los servidores-proxy. Sin embargo, como veremos en el capítulo 5 de esta memoria, esta distribución óptima no es sencilla de lograr. La distribución de los contenidos en los servidores-proxy de forma óptima exige grandes recursos de cómputo y resulta excesivamente complejo en comparación a los beneficios potenciales que se pueden obtener. Por esta razón, en la simulación se ha optado por utilizar una heurística de asignación de los contenidos que manteniendo la sencillez de implementación y bajo consumo de recursos de cómputo, puedan proporcionar unos buenos resultados. Esta asignación de contenidos implica que la distancia media de servicio finalmente obtenida en el sistema simulado sea ligeramente superior (debido a factores como la replicación de contenidos en los mirrors distribuidos a distancia-2, entre otros) a la distancia de servicio óptima. Esta diferencia justifica la desviación obtenida por la simulación respecto a los primeros resultados analíticos.

Esta desviación desaparece totalmente cuando tenemos en cuenta la distancia media de servicio real del sistema (gráficas 2 y 5). En este caso, se puede comprobar que la correlación entre los resultados de simulación y los resultados obtenidos por el modelo analítico basado en la distancia de servicio real (teniendo en cuenta la distribución de videos realmente realizada) es perfecta. Podemos ver que ambas gráficas (2, 3, 5 y 6) se superponen, independientemente de la capacidad de almacenamiento asignada a los servidores-proxy.

A partir de estos resultados podemos concluir que los resultados proporcionados por el simulador para las arquitecturas P-Tree basadas en redes no segmentadas son buenos y fiables.

4.5.2 Validación de las arquitecturas P-Tree con redes segmentadas

En este apartado vamos a validar los resultados de simulación para las arquitecturas P-Tree y Double P-Tree cuando se utilizan redes segmentados. Al utilizar estas redes segmentadas, la correlación entre la distancia media de servicio y la capacidad del sistema (ancho de banda efectivo) no se mantiene, debido a que cada enlace de la topología tiene un ancho de banda independiente. Se requiere en este caso, información del encaminamiento de los distintos flujos de datos para poder evaluar los requisitos de ancho de banda de red.

Para validar los resultados de simulación cuando se utiliza redes segmentadas vamos a utilizar el modelo analítico propuesto en el tercer capítulo (expresión 3.10). Este modelo analítico, en principio ha sido formulado teniendo en cuenta la arquitectura Double P-Tree, por lo tanto para poder obtener resultados de la arquitectura P-Tree hemos asumido una arquitectura Double P-Tree sin ningún hermano (o lo que es lo mismo una arquitectura P-Tree). La configuración utilizada ha sido un sistema con 127 redes locales, cada una con un ancho de banda de 1000 Mb/s y una capacidad de almacenamiento en los servidores-proxy para el 20% de los videos del sistema.

En la figura 4-6 se muestra el ancho de banda efectivo del sistema para las dos arquitecturas P-Tree, teniendo en cuenta la utilización de 1 ó 4 puertos de servicio (SP) y configuraciones con

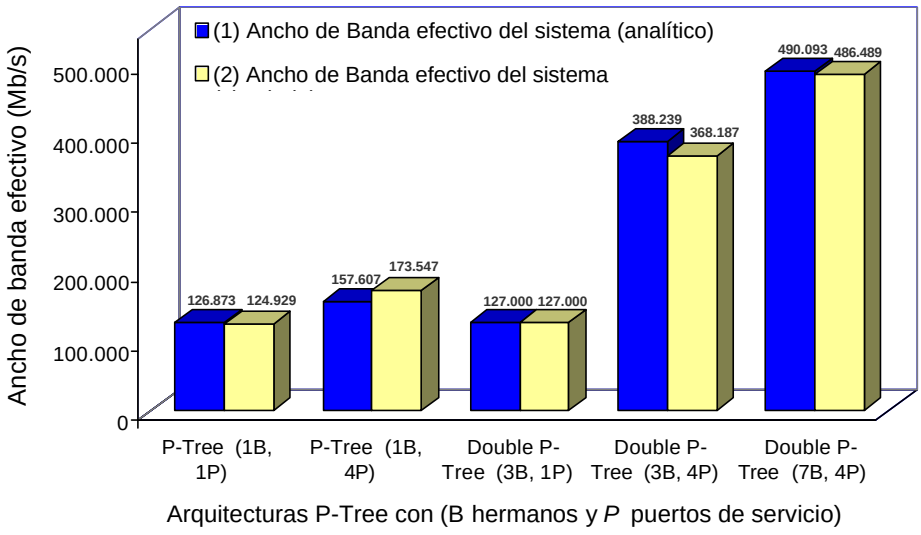


Figura 4-6. Validación de las arquitecturas Proxy-Tree y Double P-Tree utilizando redes segmentadas

3 y 7 hermanos (B) en la topología Double P-Tree. Las columnas coloreadas de azul (1) muestran los resultados obtenidos mediante el modelo analítico y las columnas en amarillo (2) muestran los resultados obtenidos mediante la herramienta de simulación.

Comparando los resultados se puede observar la buena correlación existente entre ellos. La máxima desviación entre ambos métodos se localiza en la arquitectura P-Tree con 4 puertos, en la cual existe una diferencia de apenas el 10% entre la simulación y el modelo analítico. En el caso de la arquitectura Double P-Tree, la diferencia entre los resultados de simulación y los analíticos varían desde el 0% (Double P-Tree 1B y 1P) al 5% (Double P-Tree 3B y 4P).

Las desviaciones en el rendimiento son atribuibles principalmente a la distribución de los contenidos en los servidores-proxy (el modelo analítico supone una distribución óptima) y el desbalanceo de tráfico entre los puertos de las distintas redes del sistema (el modelo analítico supone una distribución balanceada de las peticiones remotas, mientas que en la simulación esta distribución no se puede realizar de forma totalmente equilibrada).

En la figura 4-7 se muestran los anchos de banda requeridos por los servidores-proxy y las redes con mayores requisitos del sistema, debido al desbalanceo de la carga en el sistema. Estos resultados manifiestan una mayor variabilidad que los anteriores, debido a que el desbalanceo en la carga en los servidores y las redes y los servidores del sistema es uno de los principales factores que diferencian el modelo analítico y la simulación. La simulación refleja de manera más fiel el comportamiento real del sistema LVoD, lo cual implica un incremento en los desequilibrios entre los servidores y las redes debido a la distribución de los contenidos, las

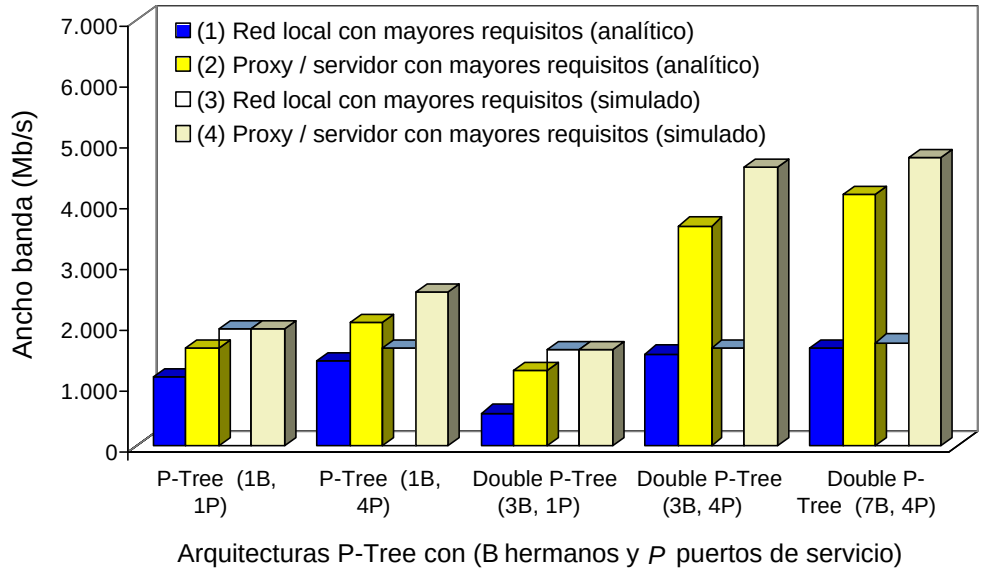


Figura 4-7. Validación de la carga de los servidores y las redes en las arquitecturas Proxy-Tree y Double P-Tree utilizando redes segmentadas.

peticiones remotas y el tráfico. A pesar de estas discrepancias entre ambos modelos, se puede advertir una amplia correlación entre los resultados obtenidos mediante la simulación (columnas trama punteada, 3 y 4) y analíticamente (columnas trama lisa, 1 y 2).

A modo de resumen, podemos concluir que la herramienta de simulación presentada en este capítulo ofrece unos resultados coherentes con los distintos modelos analíticos presentados y para las principales arquitecturas analizadas a lo largo de esta memoria (servidores-proxy de un nivel, Proxy-Tree y Double P-Tree).

Además, como se ha mostrado, el simulador nos permite analizar nuevos aspectos de los sistemas estudiados (comportamiento dinámico) que no se podían considerar mediante los distintos modelos analíticos. Estos nuevos parámetros nos abren nuevas líneas de investigación respecto a las arquitecturas propuestas. Entre estos nuevos aspectos analizados, podemos destacar las políticas de gestión del almacenamiento y el análisis del impacto del desbalanceo del tráfico en las redes sobre el rendimiento de los sistemas P-Tree. Estos dos aspectos serán estudiados en profundidad en los dos siguientes capítulos de esta memoria, mediante el uso de la herramienta VoDSim.

Capítulo 5

Gestión del almacenamiento distribuido

Resumen

La distribución del almacenamiento en los servidores-proxy entre los esquemas de caching y mirroring y la asignación de los distintos contenidos del catálogo a los distintos mirrors pueden influir en la distancia media de servicio y consecuentemente en la capacidad de servicio de las arquitecturas P-Tree. En este capítulo proponemos diversos métodos para calcular la distribución del almacenamiento de los servidores-proxy entre mirroring y caching. A continuación proponemos diversas políticas de asignación de contenidos a los mirrors del sistemas y evaluamos su influencia sobre la eficiencia final del sistema LVoD.

5.1 Introducción

El principal factor que penaliza el rendimiento de las arquitecturas distribuidas es el ancho de banda adicional requerido para atender las peticiones que no se pueden servir localmente. Mientras las peticiones que se sirven localmente consumen recursos de la red local, las peticiones que deben ser atendidas desde servidores adyacentes, requieren un ancho de banda por cada una de las redes que se tienen que cruzar.

Dada la importancia de la distancia media de servicio en los sistemas DVoD, un primer paso para mejorar la capacidad de servicio de estas arquitecturas implica reducir la distancia de servicio. La búsqueda de esta mejora se tendrá que realizar ateniéndose a los recursos disponibles (capacidad de los servidores y ancho de banda) y sin modificar las características estructurales del sistema analizado.

En las arquitecturas distribuidas de VoD la distancia de servicio se ve afectada por factores relativos a la topología de la arquitectura, el almacenamiento disponible en los servidores y la gestión de los contenidos por parte del sistema. En concreto, los parámetros más importantes a tener en cuenta son:

- ***Almacenamiento disponible en los servidores locales***

  A medida que se incrementa la capacidad de los servidores también se incrementará el número de peticiones que podrán ser atendidas localmente, reduciendo la distancia media de servicio. De forma equivalente, si reducimos la capacidad de los servidores, incrementaremos al mismo tiempo la distancia media de servicio (D_{ms}).

- ***Topología***

  La topología puede afectar de dos formas diferentes a la distancia de servicio: mediante la longitud de los caminos que se tienen que recorrer para alcanzar un determinado servidor y mediante la conectividad existente entre los distintos servidores.

  Según las características de la topología, los caminos requeridos para alcanzar un determinado servidor remoto pueden tener una longitud mayor ó menor. En función de la longitud de estos caminos, se incrementará ó se reducirá la distancia de servicio.

  Por otro lado, la conectividad es muy importante a la hora de reducir la D_{ms} cuando una petición no se puede servir localmente. Una mayor conectividad permite incrementar el número de servidores adyacentes y con ello, se incrementa la probabilidad de que un servidor vecino disponga del video requerido y no se necesite ir más lejos. En el caso de

las arquitecturas propuestas, la conectividad afecta a la capacidad del mirror distribuido a distancia-n.

- ***Distribución del almacenamiento entre los esquemas de caching y mirroring***
  Dependiendo de como se distribuya el espacio disponible en los servidores entre los dos esquemas de gestión de contenidos utilizados, la distancia de servicio puede verse afectada. Es importante realizar una correcta distribución del almacenamiento entre estos dos esquema, para poder permitir la reducción de la distancia de servicio.

- ***Asignación de contenidos a los servidores***
  La política de mapping de los videos en cada uno de los mirrors de los servidores puede influir de diversas maneras a la distancia media de servicio. Si el mapping realizado no es correcto pueden aparecer contenidos replicados en los mirrors distribuidos, provocando que determinados contenidos se tengan que ir a buscar a distancias mayores de lo estrictamente necesario e incrementado consiguientemente la distancia de servicio del sistema.

De los distintos factores que afectan a la distancia de servicio, los dos primeros, capacidad de almacenamiento y la conectividad, dependen de la configuración del sistema y variarán de una instalación a otra. La determinación de estos parámetros dependerá de políticas / decisiones ajenas a la gestión del sistema LVoD (criterios económicos, tipo de instalación, etc..). Salvo en raras excepciones, no se podrán modificar estos parámetros para mejorar el rendimiento del sistema (como mucho, se puede aconsejar a priori los valores óptimos de los mismos que permiten disponer de una mejor relación coste / eficiencia).

Si queremos mejorar la distancia de servicio, de forma independientemente de la configuración básica del sistema, debemos trabajar a nivel de políticas de distribución del almacenamiento entre caching y mirroring ó logrando políticas de asignación de contenidos a los servidores que permitan mejorar la distancia de servicio.

Este objetivo lo vamos a cubrir en los dos siguientes apartados proponiendo y analizando diversas políticas para distribuir el almacenamiento de los servidores locales (apartado 5.2) y asignar los contenidos a los mirrors (apartado 5.3).

5.2 Distribución del almacenamiento entre caching y mirroring

Las diversas arquitecturas propuestas se basan en la gestión de los contenidos del servidor mediante dos esquemas de almacenamiento diferentes: caching y mirroring. La construcción de estos sistemas se realizará siguiendo criterios de eficiencia y costes, por lo cual no es probable que los servidores dispongan de suficiente capacidad de almacenamiento como para satisfacer las necesidades máximas de ambos esquemas de gestión.

Aparece por lo tanto la necesidad de un método efectivo para poder decidir la distribución "*óptima*" del almacenamiento entre los esquemas de gestión de caching y mirroring. Tomar esta decisión no es trivial debido a que la cantidad de almacenamiento disponible en los servidores locales suele ser finita y por lo tanto cualquier modificación sobre la capacidad asignada a una de las políticas de gestión tiene una inmediata repercusión sobre la eficiencia de la otra.

El aumento de la capacidad dedicada a la cache implica que un mayor número de peticiones de los clientes podrán servirse localmente. Sin embargo, esto se realiza a costa de reducir el tamaño del mirror, provocando un incremento en la distancia media de servicio de aquellas peticiones que no puedan servirse localmente.

De forma equivalente, asignando una mayor capacidad al esquema de mirroring, reducimos la distancia de servicio de las peticiones gestionadas por los mirrors, pero también reducimos el número de peticiones servidas localmente.

Debido a esta dependencia entre la eficiencia de ambos esquemas, es necesario llegar a un compromiso entre el almacenamiento asignado a ambos esquemas de forma que al final se logre reducir la distancia de servicio del sistema de VoD.

A la hora de evaluar la eficiencia de los distintas distribuciones de la capacidad entre caching y mirroring, nos vamos a basar en la distancia media de servicio. Esta métrica es la única que de forma estática nos puede dar una visión bastante aproximada de la eficiencia del sistema con dicha distribución.

En los siguientes apartados, vamos a describir tres métodos para realizar el cálculo de la distribución óptima del almacenamiento de los servidores locales.

Los dos primeros métodos realizan una distribución basándose en un modelo analítico: el primero realiza una distribución global, asignado a todos los servidores los mismos porcentajes de cache y mirror, el segundo método adapta la distribución de forma individual para cada uno de los servidores, dependiendo de su ubicación en el árbol y la conectividad. El tercer método se basa en probar todas las distribuciones posibles y quedarse con la que ofrezca una mejor distancia de servicio.

5.2.1 Distribución global del almacenamiento de los servidores

En un primer análisis, se puede considerar que la cache debe tener preferencia debido a que los videos almacenados en ella tienen una mayor frecuencia de acceso.

Aunque la utilización exclusiva del esquema de caching puede ser eficiente para arquitecturas semi-descentralizadas (como los sistemas basados en servidores-proxy de un nivel), no es tan claro que el mismo razonamiento se pueda aplicar sin modificaciones cuando se trabaja con arquitecturas distribuidas.

Entre las razones que justifican este comportamiento diferenciado de los sistemas DVoD destacamos las siguientes:

1. Distancia de las peticiones remotas.

   En una arquitectura totalmente distribuida, no existe un servidor central que disponga de una copia completa del catálogo y, por lo tanto, las peticiones que no pueden ser atendidas por las caches de los servidores locales pueden necesitar cruzar diversas redes antes de localizar un servidor que disponga de los contenidos deseados. Esta situación provoca que si no se asigna suficiente espacio a los mirrors, las distancias de servicio serán demasiado grandes, penalizando el rendimiento del sistema.

2. Capacidad de almacenamiento finito.

   Cuando se aumenta el tamaño de la cache, al mismo tiempo se reduce la capacidad de los mirrors en los servidores, ya que la capacidad del servidor es finita y por lo tanto se cumple:

$$V_m = V_s - V_c \tag{5.1}$$

   La expresión anterior implica que el movimiento del almacenamiento de un video del mirror hacia la cache provoca que cada uno de los mirrors del sistema reduzcan su tamaño en un video.

   La reducción del tamaño de los mirrors tiene un mayor impacto sobre la eficiencia del mirror distribuido a distancia-2 (formado por el servidor local y sus vecinos inmediatos). Si asumimos un sistema con una conectividad media de C, entonces el mirror distribuido a distancia-2 sufre una reducción de C+1 videos, cada vez que se incrementa el tamaño de la cache en un contenido.

3. Nula efectividad de las caches en la gestión de peticiones remotas.

   Otro razón que no aconseja una excesiva sobreponderación del caching estriba en que los contenidos de las caches en cada uno de los servidores suelen ser idénticos. Al incrementar el tamaño de la cache también estamos incrementando el grado de replicación de videos en el sistema.

Teniendo en cuenta todos estos factores, la obtención de una distribución eficiente del almacenamiento de los servidores locales requiere de un análisis más profundo. En los siguientes apartados proponemos tres métodos para calcular esta distribución y cuyo objetivo es la reducción de la distancia de servicio del sistema de VoD.

La filosofía de los dos primeros métodos es muy parecida y se basan en la realización de tres etapas: cálculo del tamaño mínimo requerido para la cache, la evaluación del tamaño máximo del mirror que permita minimizar la replicación de contenidos, y por último, si es necesario se realiza un ajuste de la capacidad asignada inicialmente a ambos esquemas.

A continuación describiremos cada una de estas etapas y como se utilizan para implementar cada uno de los métodos.

1) Definir el tamaño mínimo de la cache

A la hora de especificar el tamaño de la cache, debemos de tener en cuenta su influencia sobre la eficiencia del mirror distribuido.

Para que el incremento de capacidad de la cache sea eficiente, la frecuencia de acceso del nuevo contenido añadido tiene que ser mayor que la suma de las frecuencias de acceso de los videos que por esta causa han sido eliminados del mirror distribuido. Esta relación sólo se cumple con videos muy populares, lo cual provoca que cuando la capacidad de almacenamiento de los servidores no es muy grande, la mayor parte sea asignada al mirror.

Una primera aproximación para calcular el tamaño mínimo de la cache, consiste en ir asignando videos mientras que estos cumplan un determinado criterio.

El primer criterio que vamos a utilizar se basa en la relación de frecuencia de acceso entre los videos de la cache y los videos del mirror distribuido a distancia-2. Se incrementará la capacidad de la cache mientras la popularidad de los contenidos añadidos sea mayor que la popularidad de los contenidos eliminados del mirror distribuido a distancia-2. Esta relación se muestra en la siguiente expresión:

$$P_i \geq (C+1) \times P_M \qquad (5.2)$$

Siendo P_i la frecuencia de acceso del video que se quiere añadir a la cache, P_M la frecuencia de acceso media de los contenidos del mirror y C la conectividad media de la topología utilizada.

Los contenidos que se quieran añadir a la cache se verificarán mediante la expresión anterior, siguiendo el orden de mayor a menor popularidad. La frecuencia de accesos a los contenidos la podemos obtener basándonos en una distribución *Zipf* ó mediante las estadísticas de accesos del sistema real. En el momento en que la expresión indique que no compensa añadir un nuevo contenido a la cache, habremos obtenido el tamaño mínimo (*i-1*) requerido para la cache.

La expresión (5.2) se concentra exclusivamente en el efecto del aumento de la capacidad de la cache sobre el mirror distribuido a distancia-2, sin tener en cuenta la influencia sobre los mirrors a distancias mayores. Mediante este sencillo criterio podemos obtener distribuciones satisfactorias para sistemas con una alta conectividad o con servidores que tengan la suficiente capacidad de almacenamiento como para evitar la aparición de peticiones que se tengan que servir a distancia–3 o superiores.

Si queremos un método que nos permita calcular el tamaño mínimo de la cache de una forma más precisa, tenemos que utilizar la distancia media de servicio para poder verificar si una determinada distribución del espacio es mejor que otra. Mediante esta nueva aproximación, iremos incrementando el tamaño de la cache mientras que la distancia media de servicio del sistema obtenida sea menor que con la configuración anterior. De esta forma el criterio que se tendría que cumplir seria el siguiente:

$$D_{ms}(V_c + 1, V_m - 1) > D_{ms}(V_c, V_m) \qquad (5.3)$$

Una vez definido, el tamaño mínimo requerido para la cache, es necesario evaluar la capacidad asignada a los mirrors. Este cálculo se realiza en el siguiente apartado.

2) Definir el tamaño máximo del mirror

Una vez hemos escogido el tamaño de la cache, el tamaño del mirror se puede deducir fácilmente utilizando la siguiente expresión:

$$V_m = V_s - V_c \qquad (5.4)$$

Sin embargo, esta primera aproximación puede dar lugar a un exceso de capacidad en los mirrors. Una sobre-dimensión de los mirrors provoca la aparición de contenidos replicados en el mirror distribuido a distancia-2. Los contenidos replicados no aportan ningún beneficio a la eficiencia del sistema, por lo tanto pueden ser eliminados sin perjudicar el rendimiento.

Para poder realizar esta redistribución del almacenamiento en los servidores, debemos definir el tamaño óptimo para los mirrors que nos permita al mismo tiempo reducir el espacio utilizado y lograr que todas las peticiones puedan ser atendidas como máximo a distancia-2. El objetivo de este análisis es ajustar la capacidad del mirror a lo estrictamente necesario para permitir almacenar una copia completa del catálogo de contenidos dentro del mirror distribuido a distancia-2.

Para comenzar este análisis debemos tener en cuenta que la capacidad del mirror distribuido a distancia-2 depende de la conectividad de la arquitectura (*C*) y de la capacidad individual de los mirrors de cada uno los servidores que componen el mirror distribuido. Por lo tanto, el tamaño mínimo del mirror se puede obtener dividiendo el número de videos que tienen que ubicarse en el mirror distribuido (tamaño del catálogo *N*, menos los contenidos previamente asignados a la cache, V_c) entre el número de servidores que conforman el mirror distribuido a distancia-2 (*C+1*).

$$V_m = \left\lceil \frac{N - V_c}{C + 1} \right\rceil \qquad (5.5)$$

Este cálculo nos indica el tamaño mínimo del mirror en videos que permite soportar todo el catálogo del sistema a distancia-2. También, nos indica la capacidad máxima que debemos asignar al mirror si no queremos que aparezcan contenidos replicados a distancia-2.

En el caso de que el tamaño obtenido para el mirror (V_m) sea mayor ó igual que la máxima capacidad disponible ($V_m \geq V_s - V_c$), entonces únicamente habrá que asignar toda la capacidad disponible al mirror, utilizando la expresión (5.4).

Si por el contrario, el tamaño máximo del mirror es menor que la capacidad máxima disponible ($V_m < V_s - V_c$), asignaremos la capacidad sobrante a la cache de los servidores. De esta forma, la capacidad finalmente asignada a la cache será:

$$V_C = V_S - V_M \qquad (5.6)$$

A continuación mostraremos la implementación del método de distribución del almacenamiento global.

Algoritmo para la distribución global

En la figura 5.1 mostramos el pseudo-código del algoritmo utilizado para calcular la capacidad de almacenamiento asignado a caching y mirroring.

```
// Paso 1: Cálculo tamaño mínimo de la cache.
MinimoCache = 0
D1 = EvaluarDistMediaServicio (0,Vs)
Do
    D2 = EvaluarDistMediaServicio (MinimoCache+1, Vs-(MinimoCache+1))
    If (D2>D1) Then
        Break
    EndIf
    MinimoCache++, D1=D2
While (MinimoCache<Vs)
// Paso 2: Cálculo tamaño máximo del mirror.
MaximoMirror = (N- MinimoCache) / (C+1)
// Paso 3: Ajustes tamaño caching y miroring
If (MaximoMirror > (Vs - MinimoCache)) Then
    Vc = MinimoCache
    Vm = Vs – Vc
Else
    Vm = MinimoMirror
    Vc = Vs - Vm
EndIf
```

Figura 5-1. Pseudo-código para la distribución almacenamiento global

En el primer paso del algoritmo se incrementa la capacidad dedicada a la cache mientras las distancia de servicio del sistema nos demuestre que la nueva configuración es mejor. La siguiente etapa calcula el tamaño máximo que deberia tener el mirror para que permita almacenar todo el catálogo en el mirror distribuido a distancia-2. Por último si el mirror no ha consumido toda la capacidad disponible en el servidor, el resto se asigna a la cache.

Utilizando el algoritmo anterior sobre una arquitectura Double P-Tree con 15 redes y con capacidad en cada servidor local para 6 de los 20 videos del catálogo, obtenemos la distribución mostrada en la figura 5-2. En la distribución obtenida se asigna el suficiente espacio a los mirrors para poder almacenar 5 videos, y el espacio restante (1 video) se asigna a la cache.

Esta capacidad es la suficiente para que dada la conectividad media de la topología utilizada (4.4 nodos vecinos), se puede almacenar un copia completa de los contenidos en el mirror distribuido a distancia 2 y reducir la distancia que necesitan las peticiones remotas para servirse.

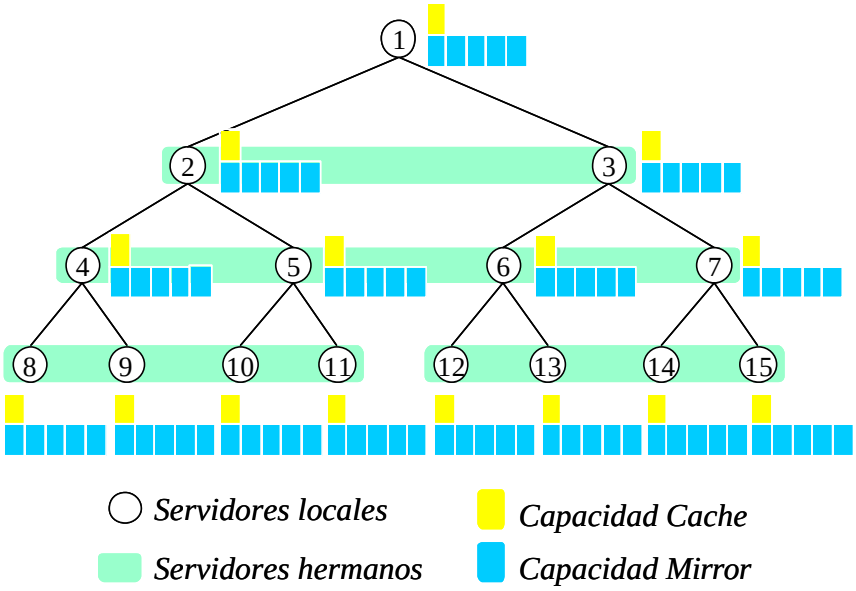


Figura 5-2. Distribución de almacenamiento Global

5.2.2 Distribución adaptiva del almacenamiento

La distribución del almacenamiento obtenida de forma global del apartado anterior, se realiza sin tener en cuenta las particularidades de cada uno de los servidores dentro del sistema de VoD. En concreto, uno de los factores que más influye a la hora de calcular la distribución del almacenamiento es la conectividad de la topología.

En el cálculo de la distribución global se utiliza la conectividad media del sistema. Sin embargo la conectividad real de cada uno de los servidores puede diferir considerablemente de un nivel de la topología a otro.

Si nos fijamos en la arquitectura ejemplo de la figura 5.2, la conectividad media del sistema es de 4.4, calculada mediante la expresión (3.9). Sin embargo, si nos fijamos en la conectividad real en los distintos niveles de la topología podemos comprobar que existen 3 grandes grupos. El primer servidor de la topología tiene únicamente 2 vecinos, los nodos de los niveles intermedios (servidores 2 al 7) tienen 6 vecinos y los nodos del último y segundo nivel tienen 4 vecinos.

La utilización de una distribución global puede provocar que los mirrors de los servidores de los niveles intermedios estén sobredimensionados (ya que al realizar la distribución se ha asumido una conectividad menor) ó que en el primer servidor su mirror sea demasiado pequeño.

La diferencia entre las conectividades medias y las reales puede afectar a la eficiencia de la distribución obtenida mediante la política global.

Para subsanar esta deficiencia, proponemos la utilización de una distribución del almacenamiento adaptada a las necesidades de cada uno de los servidores del sistema.

La adaptación de la política de distribución global a los servidores locales se consigue modificando las expresiones (5.2, 5.3 y 5.4) utilizadas para calcular el tamaño de la cache y del mirror. Estas expresiones deben tener en cuenta las características reales del sistema y no los valores promedio.

Para realizar el cálculo de la capacidad mínima de la cache adaptada a los servidores, tenemos que utilizar la distancia de servicio de cada servidor-proxy (*p*) en concreto (D_{ms}). Esta métrica mide la distancia de servicio de un servidor, teniendo en cuenta el número real de servidores que componen los distintos mirrors distribuidos a distancia-n. La única modificación que se tiene que realizar en la expresión (3.8), es la inclusión de una función que calcule el tamaño del mirror distribuido a distancia i del servidor p, $S_{MD}(p,i)$. En particular, la nueva distancia media de servicio utilizada en la expresión (5.3) se obtiene de la siguiente forma:

$$D_{ms}(p) = 1 + \sum_{i=2}^{MD} 1 - (P_c + P_{AM} * S_{MD}(p,i)) \qquad (5.7)$$

Siendo *p*, el servidor con respecto al cual se está calculando la distancia de servicio y S_{MD} la capacidad, en videos, del mirror distribuido a distancia *i* desde el servidor *p.*

Para calcular el tamaño máximo del mirror es necesario adaptar la expresión (5.5) de forma que tenga en cuenta la capacidad de los mirrors de los servidores vecinos:

$$V_M(p) = N - [V_c + S_{MD}(p,2)] \qquad (8)$$

En ambas ecuaciones, al calcular el tamaño del mirror distribuido a distancia *i* del servidor *p* se debe tener en cuenta que ocurre con los servidores sobre los cuales todavía no se ha realizado la distribución del almacenamiento. En estos casos, no se conoce todavía que distribución va a utilizar el servidor, por lo que se asumirá la capacidad del mirror ó de la cache calculada mediante la distribución global.

En la figura 5-3 se muestra el algoritmo utilizado para calcular la distribución adaptiva de almacenamiento. El algoritmo solo realiza un cálculo de la distribución por cada nivel de la topología, debido a que todos los servidores de un mismo nivel tienen la misma conectividad. Por lo tanto, solo es necesario calcular una distribución por nivel y aplicarla a todos los servidores-

```
Foreach Nivel en la topología
    V_C = CalcularCapacidadCache(Nivel)
    V_M = CalcularCapacidadMirror(Nivel)
    If ( V_M >(V_S -V_C) ) Then
        V_M = (V_S -V_C)
    Else
        V_C = (V_S –V_M)
    EndIf
    SetDistribucionServidoresNivel(Nivel, V_C, V_M)
EndForeach
```

Figura 5-3. Pseudo-código para la distribución adaptiva del almacenamiento

proxy del nivel. La complejidad de este algoritmo es del orden $O(Log_2(n))$, siendo *n* el número de servidores de la arquitectura LVoD.

Utilizando dicho algoritmo, en la figura 5-4 mostramos la distribución obtenida al aplicar la política Adaptiva. Podemos ver que las principales diferencias con respecto a la distribución Global mostrada en la figura 5-2 estriba en el tamaño de la cache y del mirror de los últimos niveles.

Se puede observar como la política asigna menos espacio al mirror de los niveles intermedios (servidores-proxy 2-7) debido a que estos nodos tienen una mayor conectividad (4 y 6 vecinos respectivamente). Asignando únicamente 3 videos al mirror, nos aseguramos la completitud del mirror distribuido a distancia-2.

En el último nivel tenemos una conectividad de 4, esto unido al hecho de que el nivel superior dedica menos espacio al mirror (3 videos) obliga al algoritmo adaptivo a asignar un video adicional al mirror de este nivel (4 videos) y únicamente dos videos al esquema de caching. Esta distribución permite tener una copia completa del catálogo en el mirror distribuido a distancia 2 de los servidores-proxy de este nivel.

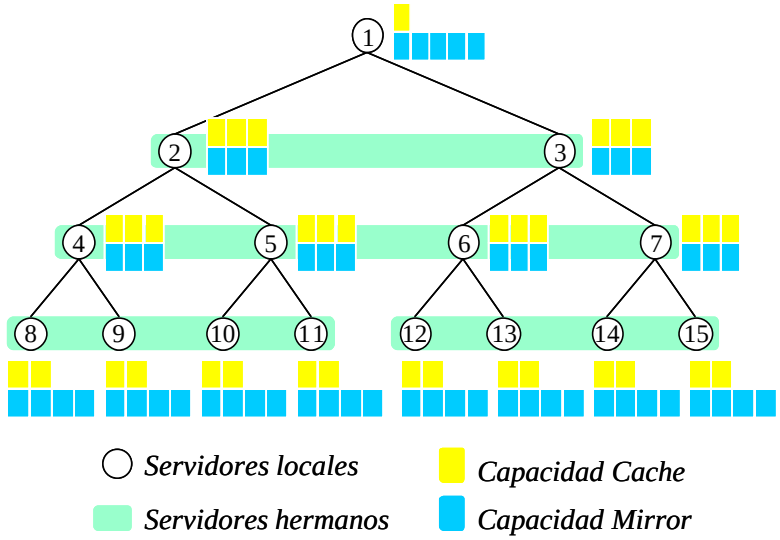


Figura 5-4. Distribución adaptiva del almacenamiento

5.2.3 Distribución óptima del almacenamiento

La última propuesta para calcular la distribución global de almacenamiento consiste en evaluar cual es la mejor de las posibles distribuciones posibles.

Este es el mismo objetivo buscado con los dos métodos anteriores. Sin embargo, al basarse éstos en la utilización de un modelo analítico para evaluar la efectividad de la distribución (distancia media de servicio), su eficacia depende en última instancia de lo representativo que sean los resultados del susodicho modelo.

En ambos casos, el modelo analítico utilizado para evaluar la distancia media de servicio (D_{ms}) no está libre de un cierto margen de error. Como ya hemos comentado en el apartado 4.5 en el capítulo 4, la D_{ms} teórica no tiene en cuenta la distribución real de los videos entre los distintos servidores-proxy ó la replicación de contenidos. Por lo tanto, la distribución del almacenamiento basada en cualquier modelo analítico puede incorporar un cierto margen de error asociado a la abstracción del sistema real realizada por el modelo analítico.

Para poder calcular la distribución global del almacenamiento que sea óptima no vamos a utilizar ninguna heurística ó modelo analítico, sino que vamos a evaluar la distancia media de servicio realmente obtenida con cada distribución propuesta.

El cálculo de la distancia media de servicio real implica la distribución del almacenamiento entre los dos esquemas de gestión y a continuación proceder a la asignación de los contenidos a cada uno de los servidores-proxy del sistema (mediante cualquiera de los algoritmos de mapping

que se presentarán en el apartado 5.3). Una vez finalizada la distribución de los contenidos ya se puede evaluar la distancia media de servicio del sistema, permitiendo compararla con las obtenidas por otras distribuciones del almacenamiento. Este proceso se repite para todas las posibles combinaciones y al final se selecciona la combinación que ofrece la mejor distancia de servicio.

El pseudo-código para la distribución global óptima se muestra en la figura 5-5. El algoritmo se basa en evaluar todas las posibles combinaciones que permite la distribución de almacenamiento entre los dos esquemas de gestión. Las distintas combinaciones se obtienen asumiendo que la unidad mínima de asignación respecto al almacenamiento es un video. Por ejemplo, si suponemos un sistema cuyos servidores-proxy tienen una capacidad para 20 videos, entonces únicamente será necesario probar 21 distribuciones diferentes (20-0, 19-1, ..., 1-19, 0-20).

Para evaluar cada una de las distribuciones primero se debe inicializar el almacenamiento de cada uno de los servidores, asignando el espacio correspondiente a los esquemas de caching y mirroring. A continuación se realiza la asignación de los videos al mirror y a la cache de cada servidor del sistema. Finalmente se evalúa la distancia de servicio actual y si ésta es menor se actualiza la configuración óptima.

El principal inconveniente de este método es el mayor tiempo de cómputo requerido para poder decidir la distribución del almacenamiento. La complejidad de este algoritmo es del orden O(n), siendo n la capacidad del servidor-proxy en número de videos.

```
MejorDms=-1
For (CapacidadMirror=Vs; CapacidadMirror>=Vs; CapacidadMirror--)
    InicializarAlmacenamiento(Vs-CapacidadMirror, CapacidadMirror)
    AsignarContenidosMirrors()
    AsignarContenidosCaches()
    Dms = CalcularDistanciaMediaServicio()
    If (Dms<MejorDms or MejorDms<0) Then
        MejorDms = Dms
        VM = CapacidadMirror
        VC = Vs - CapacidadMirror
    EndIf
EndFor
```

Figura 5-5. Pseudo-código para la obtención de la distribución óptima de almacenamiento

El tiempo de cómputo de este algoritmo se puede reducir mediante la utilización de métodos de búsqueda más sofisticados (búsqueda binaria) dentro del espacio de soluciones (distribuciones posibles). Esto nos permitiría reducir el número de combinaciones evaluadas a únicamente $\log_2(V_s)+1$.

La figura 5-6 muestra la distribución global óptima obtenida al aplicar este método. En este caso se asigna de forma global 4 videos al esquema de gestión de mirroring y 2 videos a la cache. Esta distribución del almacenamiento permite que todos los nodos de servicio (salvo el situado en el primer nivel de la topología) formen parte de un mirror distribuido a distancia-2 con suficiente capacidad para almacenar todos los videos del catálogo del sistema LVoD. Simultáneamente, se aumenta el porcentaje de peticiones que pueden ser gestionadas localmente gracias a la asignación de 2 videos para la cache.

5.2.4 Evaluación de las políticas de distribución del almacenamiento

En este apartado evaluaremos las tres políticas de distribución del almacenamiento propuestas en los apartados anteriores. Los resultados mostrados en este capítulo se obtienen a partir de la utilización de la herramienta de simulación descrita en el cuarto capítulo.

Para evaluar estas políticas asumiremos la utilización de una arquitectura Double P-Tree compuesta por 63 servidores-proxy con 3 puertos de servicio y un ancho de banda de servicio 300 Mb/s. El sistema utilizará 63 redes locales basadas en switches segmentados con un ancho de banda de 100 Mb/s y una topología con 3 servidores hermanos

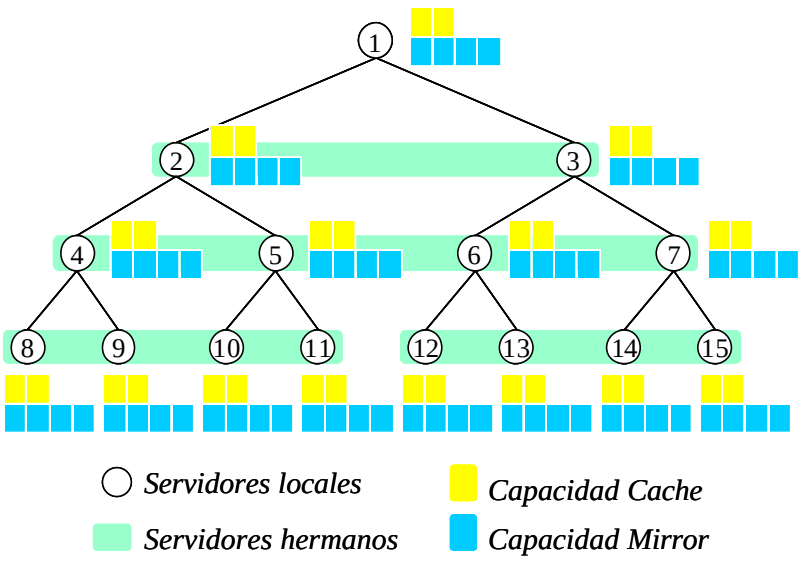


Figura 5-6. Distribución óptima de almacenamiento global

La evaluación de la efectividad de las distintas políticas se realizará utilizando como métrica principal el ancho de banda efectivo del sistema, o lo que es lo mismo, el número de usuarios que éste puede atender de forma independiente.

Para poder evaluar la capacidad de servicio del sistema es necesario saturar el sistema hasta que alcance su máxima capacidad de servicio (pero sin provocar rechazos) y evaluar en ese instante el número de usuarios activos. La saturación del sistema se logra utilizando una frecuencia de llegada de peticiones alta (en concreto 10 peticiones por minuto en cada una de las redes locales) y obteniendo una instantánea del estado del sistema cuando los requisitos de ancho de banda de red del sistema alcanzan el máximo permitido (63 redes * 100 Mb/s).

La figura 5-7 muestra la efectividad de cada una de las políticas de distribución del almacenamiento para distintas configuraciones de los servidores-proxy. En concreto mostramos los resultados para sistemas en los cuales los servidores-proxy pueden almacenar desde el 5% al 40% de los contenidos del catálogo disponible en el sistema.

Los resultados muestran que cuando se dispone de poco almacenamiento en los servidores-proxy (inferior ó igual al 15%) el método de distribución adaptiva supera ligeramente al resto de métodos.

Sin embargo, cuando los servidores tienen la suficiente capacidad de almacenamiento la conectividad de los nodos en los distintos niveles de la topología, ya no afecta de forma tan importante al rendimiento de la distribución. Por lo tanto, con servidores-proxy que disponen de una capacidad superior ó igual al 20%, el método que obtiene los mejores resultados es la

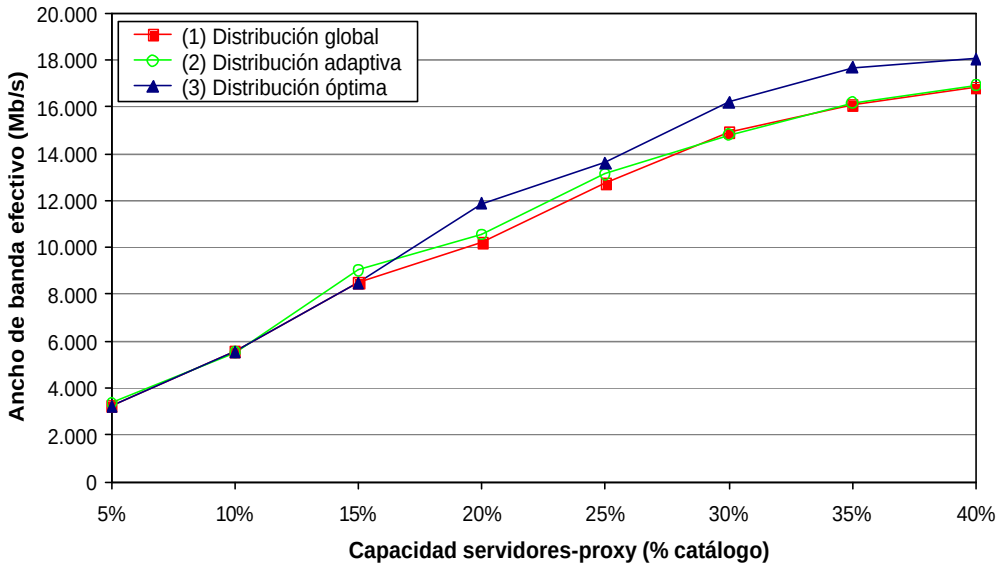


Figura 5-7. Rendimiento de las políticas de distribución del almacenamiento

distribución global óptima. Esta distribución obtiene en promedio una mejora del 12% con respecto a los otros métodos alternativos.

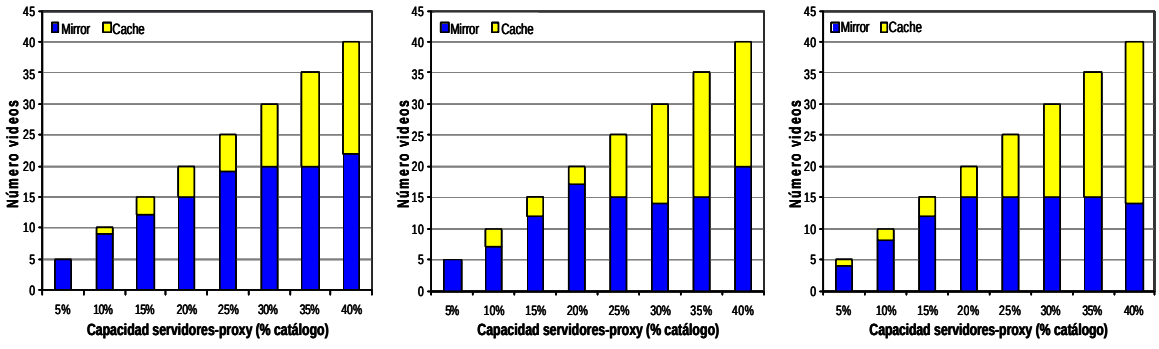
De los distintos métodos, el que obtiene los peores resultados es la política global basada en un modelo analítico. La efectividad de esta distribución, al igual que la distribución adaptiva, se basa en la exactitud del modelo analítico para evaluar la distancia media de servicio del sistema. Los errores que se heredan de la simplificación que representa todo modelo analítico afectan a la efectividad de ambos métodos.

Por otro lado, el método de distribución óptimo se basa en la distribución de videos reales y por lo tanto, no sufre los problemas de las distribuciones basadas en modelos analíticos y permite obtener globalmente los mejores resultados. Sin embargo, debido a que la distribución óptima trabaja a nivel global (la misma distribución para todos los servidores-proxy del sistema) no le permite obtener siempre los mejores resultados.

En la siguiente figura (5-8) podemos ver las distintas distribuciones de almacenamiento obtenidas para cada uno de los métodos. En estas distribuciones podemos constatar el papel preponderante que tiene el esquema de mirroring sobre la eficiencia de los sistemas P-Tree.

Si nos fijamos en las distribuciones óptimas (figura 5-8a) podemos observar que cuando el espacio de almacenamiento en los servidores-proxy es pequeño (5%-25%), la mayoría de la capacidad del servidor (al menos el 75%) se dedica al mirror. A partir del instante en que la capacidad de almacenamiento de los servidores-proxy supera el 25% se incrementa progresivamente la capacidad asignada a la cache hasta alcanzar una distribución paritaria entre caching y mirroring.

La principal diferencia existente entre las distribuciones obtenidas mediante el método óptimo (figura 5-8a) y las distribuciones obtenidas con los métodos basados en un modelo analítico



(a) Distribución óptima **(b) Distribución global** **(c) Distribución adaptiva**

Figura 5-8. Distribución de la capacidad de almacenamiento en los servidores-proxy

(distribución global y distribución adaptiva, figura 5-8b y figura 5-8c) consiste en que estas últimas le dedican un porcentaje mayor del almacenamiento al esquema de caching. Este mayor porcentaje asignado a la cache se produce porque el modelo analítico penaliza en menor medida las peticiones remotas (ya que no se tiene en cuenta la influencia de la replicación de contenidos entre los mirrors).

Estos resultados ponen de manifiesto la importancia de utilizar ambos esquemas de gestión para los contenidos de los servidores-proxy. La utilización exclusiva del esquema de caching (al igual que utilizan los sistemas de un nivel de servidores-proxy) implicaría una saturación del servidor principal, mientras que la utilización de únicamente el esquema de mirroring implicaría una reducción del rendimiento del sistema causada por la reducción del número de peticiones atendidas localmente.

5.3 Asignación de los videos en los mirrors distribuidos

La forma en que se distribuyen los videos entre los mirrors de los servidores locales puede afectar también a la distancia media de servicio y, por lo tanto, a la capacidad de streaming del sistema de VoD.

Conseguir una distribución óptima es considerablemente complejo debido al gran número de combinaciones que se pueden obtener. El problema de la asignación de los videos en sistemas de VoD jerárquicos ha sido estudiada en la literatura con el objetivo primordial de minimizar los costes de transmisión por la red [Ouy98] [Hwa01] [Zho02] ó bien para optimizar la calidad de servicio (entendida en este caso como el factor de compresión del video) ofrecida al usuario [Zho00].

Muchas de estas soluciones se basan en aproximaciones que realizan búsquedas exhaustivas de la solución óptima dentro del espacio de soluciones generado en base a una función de coste. Las soluciones aportadas por estas aproximaciones requieren elevados tiempos de cómputo y no se ajustan a las características especificas de las arquitecturas P-Tree (una arquitectura totalmente distribuida, que utiliza simultáneamente caching y mirroring distribuido).

En nuestro caso, para reducir la complejidad del algoritmo y los requisitos de cómputo, hemos propuesto dos nuevas políticas de asignación de los videos a los mirrors basadas en la utilización de distintas heurísticas. El objetivo de estas políticas es reducir la distancia media de servicio del sistema, teniendo en cuenta factores tales como: la completitud, la replicación de contenidos y los efectos colaterales entre servidores.

- ***Completitud***

  El concepto de completitud hace referencia a que siempre que haya suficiente almacenamiento disponible en el sistema ó en un determinado mirror distribuido, el algoritmo de asignación debe lograr mapear, al menos, una copia completa del catálogo de contenidos.

  Siguiendo este criterio, la política de mapping debería poder asegurar que cada uno de los videos del catálogo se encuentra en al menos alguno de los mirrors del sistema (*Completitud de contenidos en el sistema*).

  De forma equivalente, siempre que el mirror distribuido a distancia-N tenga suficiente capacidad como para almacenar todo el catálogo, el mapping realizado debería lograr que todos los videos pudiesen ser accedidos como máximo a esa distancia. (*Completitud de contenidos en el mirror distribuido a distancia-n*)

- ***Replicación de contenidos***

  Intentar reducir el número de videos replicados en el mirror distribuido a distancia-2. Cuando el mirror distribuido a distancia-2 no tiene suficiente capacidad para almacenar todo el catálogo, la aparición de replicas afecta al redimiendo del mapping ya que implica que aleja contenidos que se podrían encontrar más cerca.

- ***Efectos colaterales***

  La asignación realizada en los servidores puede influir negativamente en los mappings de los servidores vecinos. Por ejemplo, si se realiza la asignación de los videos a un servidor atendiendo a criterios puramente locales y sin tener en cuenta a los servidores vecinos, puede producirse la aparición de contenidos replicados ó que determinados contenidos se localicen a distancias largas.

  Por lo tanto, se requiere un análisis global de la distribución de los contenidos para poder reducir los efectos colaterales.

5.3.1 Mapping secuencial

Esta primera política de mapping propuesta es muy sencilla y básicamente nos servirá para detectar los problemas que pueden surgir al realizar la asignación de los videos y como base de comparación para las siguientes propuestas.

Esta política se basa en asignar los contenidos multimedia a los mirror de forma secuencial. La política funciona de la siguiente forma, se recorren los servidores-proxy de la topología en un

determinado orden y al primer servidor se le asignan los primeros Vm videos (siendo V_m el número de videos que caben en almacenamiento asignado para mirroring), al segundo servidor los siguiente V_m videos y así sucesivamente hasta completar todos los servidores del sistema. Una vez se asigna el último contenido del catálogo del sistema se continua por el primero.

Se ha preferido realizar una distribución de los videos secuencial frente a una distribución aleatoria debido a que ésta última no asegura la completitud a nivel de sistema. Con lo cual, podría darse el caso de que una vez finalizada la asignación, el sistema no dispusiese de al menos una copia de cada uno de los contenidos del catálogo del sistema.

El pseudo-código del mapping secuencial se describe en la figura 5-9. Primeramente, inicializamos aleatoriamente a partir de que contenido se empieza la distribución secuencial. Seguidamente se recorre cada uno de los servidores del sistema de VoD, asignando cada uno de ellos los siguientes V_m videos que se almacenarán en el espacio dedicado al mirroring.

En la figura 5-10, mostramos la distribución de los contenidos en el sistema, utilizando la política de asignación secuencial. Para todos los ejemplos utilizados en este apartado se asume una arquitectura Double P-Tree con 15 redes locales, en la cual cada servidor-proxy tienen capacidad para 6 videos (4 asignados a miroring y 2 a caching, obtenidos a partir de la distribución óptima) de los 20 que conforman el catálogo del sistema. Para realizar el cálculo de la distancia media de servicio más ajustado a la realidad, se tiene que tener en cuenta todos los contenidos de los servidores-proxy (caching+mirroring). Por lo tanto, en todos los ejemplos, después de asignar los contenidos a los mirrors, se inicializa los contenidos de las caches de cada uno de los servidores. Para realizar esta inicialización se asignan los videos más populares que no hayan sido previamente asignados al mirror. En este caso, estamos suponiendo una

```
videoId = Randon(1,Vs)
Foreach ServidorLocal
    For m=0 To M-1
        ServidorLocal.SetContenidoMirror(videoId)
        If (videoId>N) Then
            videoId = 1
        EndIf
    EndFor
EndForeach
```

Figura 5-9. Pseudo-código de la política de asignación secuencial

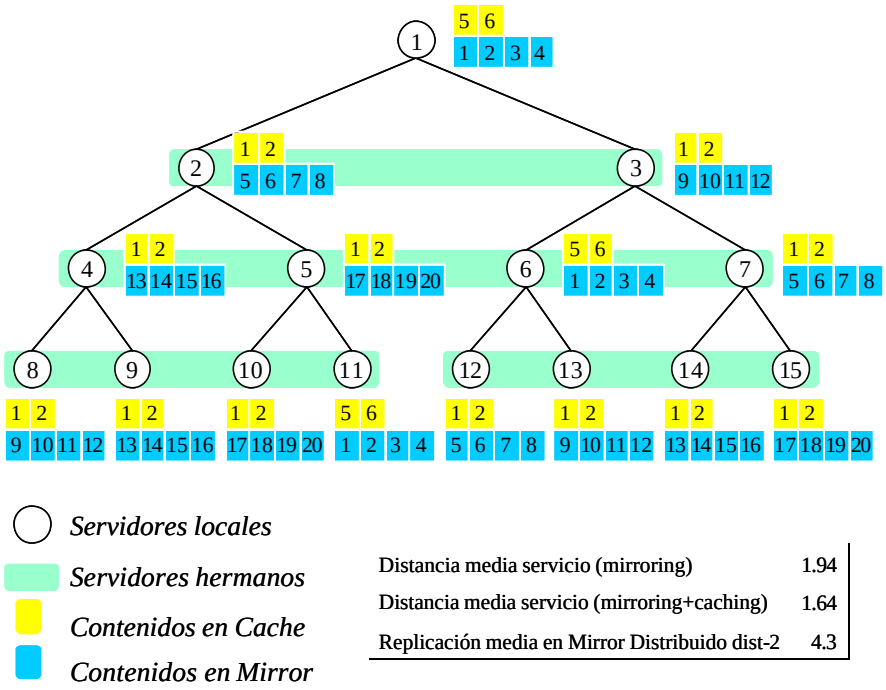


Figura 5-10. Asignación de videos obtenida con el mapping secuencial

distribución de la popularidad de los videos *Zipf* y que los videos más populares son aquellos con un identificador menor.

En la figura 5-10 podemos ver la distribución de videos en los mirrors obtenida al aplicar la política secuencial. Debido a su sencillez, la asignación secuencial no permite aprovechar adecuadamente (con respecto a la distancia media de servicio) el almacenamiento disponible en los servidores del sistema. Se puede observar que la asignación finalmente obtenida no cumple ninguno de los criterios descritos con anterioridad. Por ejemplo: en el servidor número 5, a pesar de tener un mirror distribuido a distancia-2 con capacidad para 28 videos no cumple la condición de completitud ya que los videos 13-16 (almacenados en el servidor número 9) no pueden ser accedidas a distancia 2. Además, el mirror distribuido a distancia 2 de dicho servidor replica varias veces los videos 1-8 y 17-20.

5.3.2 Mapping ByDistance

Para mejorar la asignación obtenida con el mapping secuencial debemos tener en cuenta los contenidos ya asignados en los servidores vecinos. Si tenemos en cuenta esta información y

utilizando un determinada heurística se tomarán las decisiones sobre que contenidos se van asignando a cada uno de los servidores del sistema de VoD.

Recordemos que nuestro objetivo es reducir la distancia de servicio. Este parámetro se ve considerablemente penalizado por los contenidos más alejados que requieren el cruzar diversas redes para poder acceder a ellos. Por lo tanto, una buena política de mapping de conseguir acercar aquellos contenidos que se encuentran más lejos de los servidores.

La heurística que utilizaremos tiene en cuenta el planteamiento anterior y por lo tanto a la hora de realizar el mapping escogerá aquellos contenidos que o bien aún no están disponibles en el sistema (porque todavía no han sido asignados a ningún servidor) ó bien aquellos contenidos que se encuentran más alejados del servidor que se está procesando. Con esta heurística intentamos cumplir el criterio de completitud anteriormente descrito.

Mediante la utilización de esta heurística obtenemos una nueva política para la asignación de los contenidos a los mirrors denominada ByDistance debido a que utiliza como función de coste la distancia de servicio [Cor03][Cor03b]. El pseudo-código correspondiente se muestra en la figura 5-11.

El algoritmo, antes de realizar la asignación en cada uno de los servidores, calcula la distancia a la que se encuentran cada uno de los contenidos que conforman el catálogo del sistema. Se crea una tabla de distancia, en la cual se asigna a cada uno de los contenidos la menor distancia a la cual se puede acceder a ese contenido desde el servidor que se está procesando. Si algún contenido aún no se encuentra localizable en algún mirror, se marca como no asignado. El algoritmo utiliza la información de esta tabla de distancias para seleccionar los videos que se asignarán al mirror.

```
Foreach ServidorLocal
    CalcularDistanciaContenidos(ServidorLocal)
    For i=0 To M-1
        videoId = ObtenerContenidoMásAlejado()
        ServidorLocal.AsignarContenidoMirror(videoId)
    EndFor
EndForeach
```

Figura 5-11. Pseudo-código de la política de asignación ByDistance

La heurística, a la hora de seleccionar los videos, se guía por dos criterios básicos:

1. Se intenta seleccionar todos aquellos contenidos que todavía no han sido asignados a ningún mirror.

   De esta forma se intenta cumplir el criterio de completitud y que el sistema de VoD al menos tenga una copia de todos los videos que conforman el catálogo.

2. Mientras todavía quede espacio en el mirror del servidor, se asignan los videos que estén más alejados. En el caso de que exista más de un video igualmente alejados, la selección de cual de ellos es el escogido se puede realizar de forma secuencial ó bien aleatoriamente.

La figura 5-12 muestra el proceso de decisión del algoritmo de mapping en un servidor intermedio, en concreto la asignación del segundo contenido al servidor número 9. Podemos ver la tabla de distancias de los distintos contenidos, la cual se utiliza como base para tomar la decisión de cual será el siguiente video que se asignará al mirror del servidor.

La distribución de los vídeos finalmente obtenida se muestra en la figura 5-13. Podemos observar que gracias a la heurística se consigue reducir la distancia media de servicio (pasando de un 1.64 a un 1.59), mientras que se mantiene el grado de replicación de contenidos en los mirrors de los servidores adyacentes (de media se replican 4.3 videos en el mirror distribuido a distancia-2).

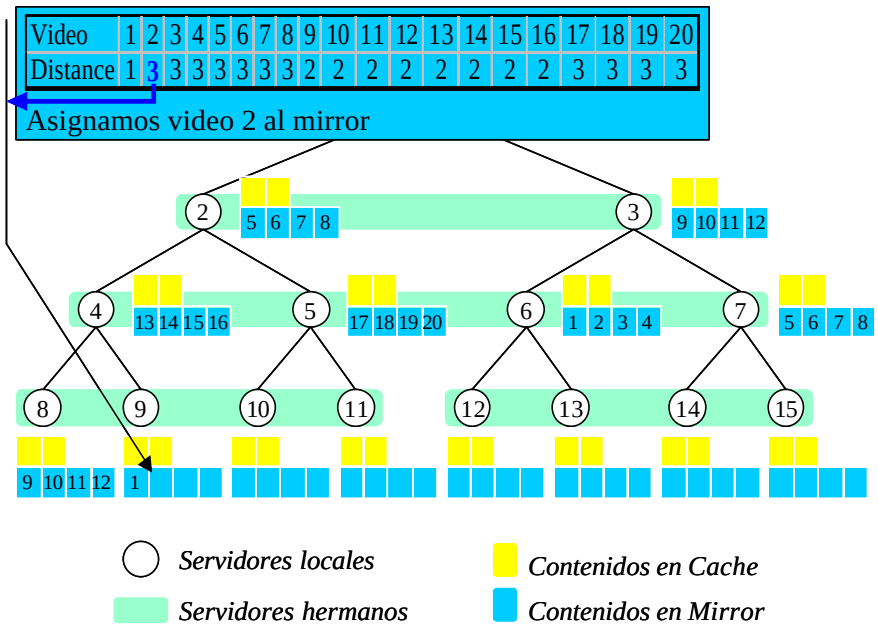


Figura 5-12. Paso intermedio de decisión en el algoritmo ByDistance

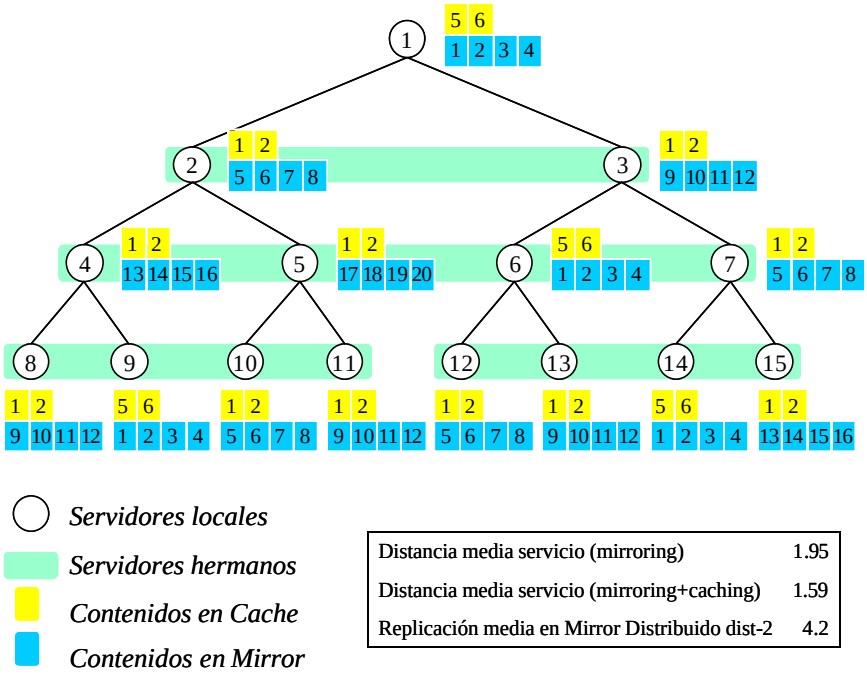


Figura 5-13. Asignación de videos obtenida con el mapping ByDistance

5.3.3 Mapping ByDistance optimizado

Otro problema que surge a la hora de distribuir los contenidos entre los mirrors deriva de los efectos colaterales que puede producir una determinado mapping en un servidor sobre los servidores vecinos.

El principal inconveniente de los efectos colaterales es la aparición de contenidos replicados en el mirror distribuido a distancia 2 a costa de otros contenidos que no se pueden localizar a distancia 2 ó inferior.

Para atenuar este problema, pero manteniendo al mismo tiempo la heurística para reducir la distancia media de servicio, vamos a modificar el algoritmo anterior, cuyo funcionamiento básico se sigue manteniendo.

A la hora de seleccionar los videos se mantienen los dos criterios vistos anteriormente, pero en el caso de que se pueda escoger entre distintos contenidos que cumplan el mismo criterio, entonces se utilizará un tercer criterio que intentará reducir los efectos colaterales. En este caso se tendrá en cuenta como afecta la asignación de cada uno de los contenidos a los servidores

vecinos, comprobando si dicho contenido esta ya replicado en el mirror distribuido a distancia-2 del servidor vecino. En caso afirmativo, se escogerá aquel contenido que provoque menos replicación en los mirrors vecinos.

El algoritmo con esta optimización será idéntico a los mostrados en las figuras 5-9 y 5-11. Únicamente se tendrá que modificar la función que devuelve el contenido seleccionado para ser asignado al mirror. Esta función tendrá que tener en cuenta el criterio de replicación en los servidores vecinos en el caso de que hayan más de un video candidato.

En la figura 5-14 mostramos la asignación de videos obtenida mediante la política de asignación ByDistance, intentando reducir los efectos colaterales. En los resultados mostrados en el ejemplo se puede constatar que la distancia media de servicio es similar a la obtenida mediante el método ByDistance original (1.59).

Sin embargo si nos fijamos en la replicación de videos en el mirror distribuido a distancia-2, podemos comprobar que la replicación se reduce en un 20%, pasando de 4,3 videos replicados a únicamente 3,4. Esta reducción en la replicación va a permitir que esta política de asignación logre mejorar los resultados de la política ByDistance cuando se tenga en cuenta el rendimiento de los sistemas P-Tree, en el apartado 5.3.5.

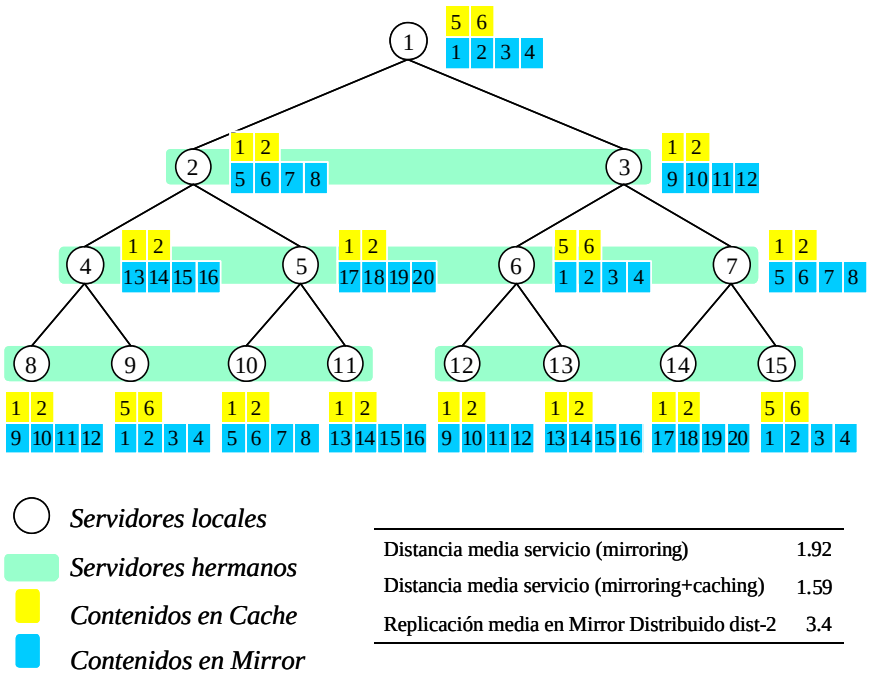


Figura 5-14. Asignación de videos obtenida con el mapping ByDistance optimizado

5.3.4 Orden de procesamiento de los servidores-proxy

Otro parámetro que podría afectar al rendimiento de las políticas de mapping es el orden en que se procesan los servidores de VoD al realizar la asignación de los videos al mirror.

Se han probado diferentes secuencias (Pre-orden, Post-orden y por niveles) de procesamiento de los servidores en el árbol de la topología. Las distintas secuencias de procesamiento surgen de los distintos métodos para recorrer una topología en árbol:

- **Por niveles:**

  Mediante este método los servidores-proxy se procesan siguiendo los distintos niveles de la topología. Primero se procesan todos los servidores del primer nivel de la topología, a continuación todos los servidores del segundo nivel y así sucesivamente hasta que se alcanza el último nivel del árbol y se procesan todos sus nodos.

  La figura 5-15a se muestra el orden de procesamiento por niveles de los servidores-proxy de una arquitectura Double P-Tree compuesta por 15 redes locales.

- **Pre-orden:**

  Mediante este recorrido primero se procesa el servidor-proxy origen, después se procesa su hijo izquierdo y a continuación su hijo de la derecha. Este orden se realiza de forma recursiva hasta que se han procesado todos los nodos de servicio del árbol. La figura 5-

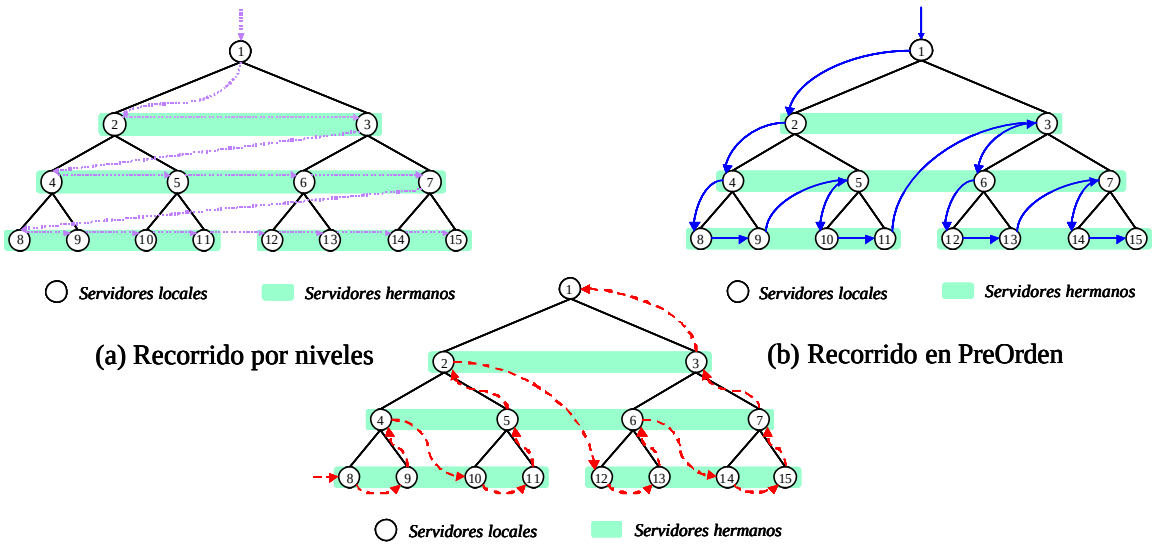


Figura 5-15. Diferentes órdenes de procesamiento de los servidores-proxy al realizar la asignación de los videos

15b muestra el orden de procesamiento de los servidores al aplicar un recorrido Pre-orden.

- **Post-orden:**
  Mediante este orden de recorrido, los servidores-proxy hijos tienen preferencia frente a los servidores raíz. Primero se procesa el servidor-proxy hijo de la izquierda, después se procesa el hijo de la derecha y por último el servidor padre. Al igual que con el resto de órdenes, este recorrido se realiza de forma recursiva hasta que se han procesado todos los nodos de servicio del árbol. La figura 5-15c se muestra el orden Post-orden de procesamiento en una arquitectura Double P-Tree.

La característica más importante que se debe tener en cuenta a la hora de seleccionar un método de recorrido es que permita procesar de forma consecutiva todos los servidores-proxy que constituyen un mirror distribuido a distancia-2, de forma que se reduzcan los efectos colaterales. Sin embargo, esta característica es imposible de cumplir totalmente debido a que los distintos mirrors distribuidos se encuentran solapados entre si.

Por lo tanto, nos hemos de decantar por el recorrido que más se aproxime. En el caso de los órdenes propuestos el que peor cumple estos requisitos es el recorrido Pre-orden. Mientras, los otros dos métodos de recorrido intentan cumplir con este criterio ya sea procesando conjuntamente todos los servidores-proxy hermanos (por niveles) ó procesando el padre y los hijos de forma consecutiva (Post-orden).

5.3.5 Evaluación de las políticas de asignación

A continuación vamos a evaluar el rendimiento de las distintas políticas de asignación de videos presentadas en este capítulo. Al realizar esta evaluación hemos supuesto un sistema Double P-Tree compuesto por 63 servidores-proxy con una ancho de banda de servicio de 300 Mb/s distribuido entre 3 puertos de servicio. Las redes locales utilizan switches segmentados con un ancho de banda de 100 Mb/s.

Políticas de asignación

En la figura 5-16 comparamos la capacidad de servicio (ancho de banda efectivo) obtenida al utilizar las distintas políticas de asignación de videos (Secuencial, ByDistance y ByDistance optimizado). Para analizar el rendimiento de estas políticas de forma independiente al almacenamiento disponible en los servidores, se ha probado cada una de las políticas con distintas capacidades de almacenamiento (5%-50%) en los servidores-proxy del sistema.

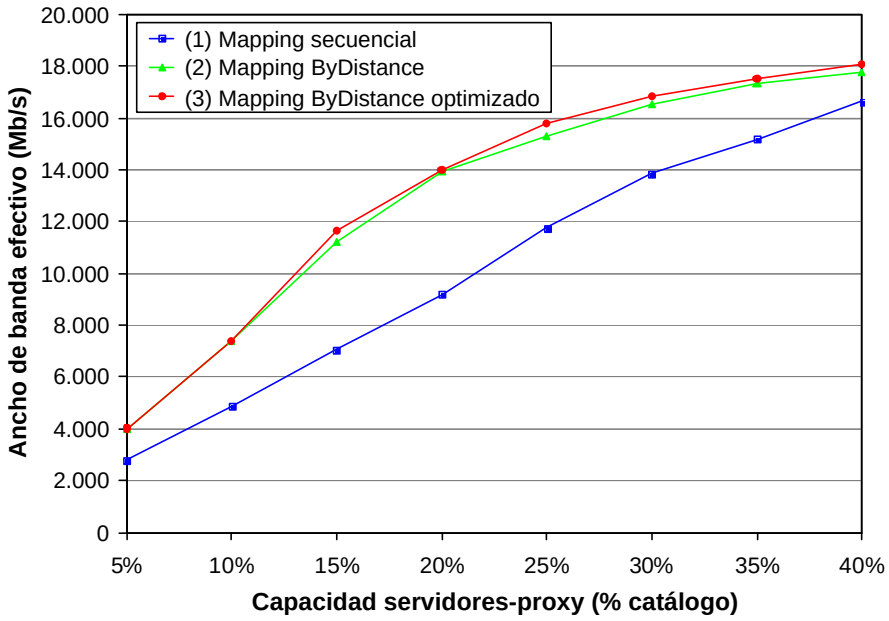


Figura 5-16. Rendimiento políticas de asignación de videos a los mirrors

Mediante estos resultados podemos afirmar que las dos variantes de la política de asignación ByDistance (gráficas 2 y 3) superan ampliamente el rendimiento obtenido por la política secuencial. La capacidad de servicio del sistema puede aumentar hasta en un 60% (11200 Mb/s en contraposición a los 7000 Mb/s con la asignación secuencial) cuando el almacenamiento disponible en los servidores-proxy es pequeño (rango 5-20%). Precisamente, es bajo estas condiciones (cuando no hay suficiente capacidad en los mirrors distribuidos a distancia-1 para mantener una copia completa del catálogo) cuando las políticas de asignación deben demostrar sus cualidades.

A medida que aumentamos la capacidad de almacenamiento de los servidores-proxy, la política de asignación ejerce una menor influencia sobre el rendimiento del sistema. En estos casos la diferencia de rendimiento entre la política secuencial y las políticas ByDistance se reduce progresivamente (proporcionando un incremento de prestaciones del 30%, 20% 15% y 9% cuando la capacidad de los servidores-proxy es del 25%, 30%, 35% y 40% respectivamente).

Comparando las dos variantes de la política ByDistance entre si, podemos constatar que la política que tiene en cuenta los efectos colaterales obtiene siempre unos resultados superiores ó similares a los obtenidos por la política ByDistance original. La diferencia de prestaciones entre ambas políticas se sitúa alrededor del 10%. A partir de este instante, siempre que hagamos referencia a la política ByDistance se hará referencia a la política optimizada.

Como consecuencia de estos resultados, podemos concluir que, a pesar de su sencillez de implementación, las políticas de mapping ByDistance son una alternativa más que aceptable para la asignación de contenidos en arquitecturas LVoD distribuidas, logrando incrementos considerables en la efectividad del sistema.

Influencia del orden de procesamiento de los servidores

La figura 5-17 muestra la influencia sobre la capacidad de servicio del sistema LVoD del orden de procesamiento de los servidores-proxy a la hora de realizar la asignación de los contenidos a los mirrors.

De los distintos órdenes de procesamiento, los mejores resultados se obtienen mediante los recorridos en Post-orden y por niveles. El primero obtiene mejores resultados cuando la capacidad de los servidores-proxy es mayor (rango 30%-40%), mientras que el recorrido por niveles destaca cuando el almacenamiento es menor (rango 5%-25%).

La comparación de estos resultados con los obtenidos cuando el árbol se recorre en Pre-orden nos demuestra la influencia del orden en que se realiza el procesamiento de los servidores a la hora de realizar la asignación de los videos sobre el rendimiento del sistema. El recorrido en Pre-orden penaliza la capacidad de servicio hasta un 18% y siempre obtiene peores resultados que los otros dos tipos de recorrido.

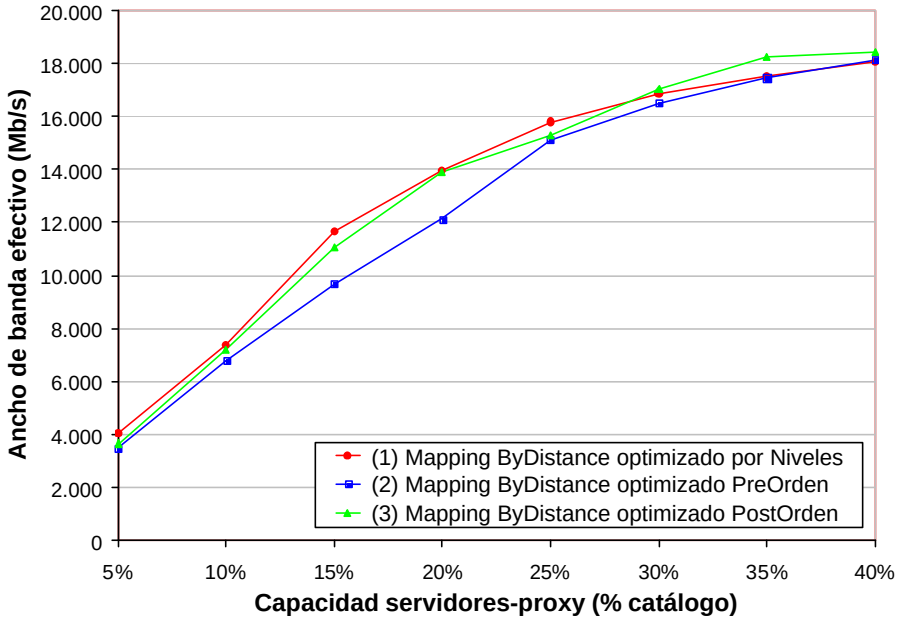


Figura 5-17. Rendimiento política de asignación ByDistance en función del orden de procesamiento de los servidores-proxy

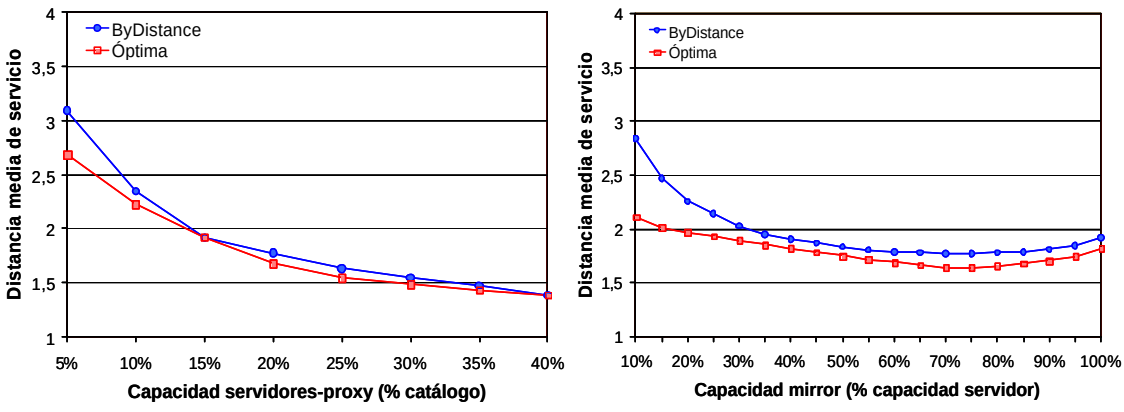
Asignación ByDistance & asignación óptima

Para finalizar la evaluación de las políticas de asignación de los contenidos vamos a evaluar la efectividad de la heurística de asignación de videos ByDistance respecto a una política de asignación óptima.

La realización de este análisis implica evaluar el rendimiento de una política de asignación de videos óptima. En este estudio vamos a utilizar la distancia media de servicio teórica calculada en el capítulo 3 (mediante la expresión 3.8) para aproximar el rendimiento obtenido por una distribución de videos óptima. Sin embargo, hay que tener en cuenta que la distancia media de servicio obtenida a partir de esta expresión difícilmente podrá ser alcanzable por cualquier asignación de videos debido a la replicación de contenidos y a la suposición de una conectividad homogénea en toda la topología.

En la figura 5-18a se compara la distancia de servicio óptima con la distancia media de servicio realmente obtenida al aplicar el mapping ByDistance. Podemos observar que el mapping ByDistance de media obtiene una distancia media de servicio un 8% superior a la distancia media de servicio que se obtendría mediante una política de asignación óptima. Como ya hemos comentado una parte de este menor rendimiento puede ser atribuido al sobreoptimismo en el cálculo de la distancia de servicio teórica, mientras que el resto se puede atribuir a la mejora de rendimiento que obtendría una distribución de videos óptima.

En la figura 5-18b se muestra la distancia media de servicio óptima y la obtenida por la política de asignación ByDistance para una arquitectura Double P-Tree cuyos servidores tienen una capacidad de almacenamiento para el 20% de los contenidos del catálogo y variando el porcentaje de esta capacidad dedicada al mirror. Se puede observar que en el rango más



(a) Capacidad del servidor **(b) Distribución del almacenamiento**

Figura 5-18. Distancia media de servicio óptima versus distancia de servicio ByDistance

significativo (40%-80% de capacidad del servidor-proxy dedicada al mirror) la diferencia de rendimiento de la heurística es inferior al 10%.

A partir de estos resultados podemos concluir que la asignación de videos realizada por la heurística ByDistance logra una buena distancia de servicio sin requerir una alta complejidad, del orden de o(n), siendo n el número de servidores del sistema.

5.4 Conclusiones

El análisis realizado en este capítulo ha permitido poner de manifiesto la importancia de la gestión del almacenamiento de los servidores que conforman un sistema distribuido de VoD. En concreto se han analizado dos aspectos relacionados con el almacenamiento: el porcentaje de almacenamiento de los servidores-proxy dedicado a cada uno de los esquemas de gestión de contenidos (caching / mirroring) y la asignación de los videos a los mirrors de los servidores-proxy del sistema.

Los resultados obtenidos a lo largo del capítulo ponen de manifiesto que la utilización en exclusiva del esquema de caching (tal y como proponen las arquitecturas basadas en servidores-proxy de un nivel) ó solo el esquema de mirroring, no son por si solos una buena solución. Las arquitecturas distribuidas de LVoD (que no repliquen todos los contenido en todos los nodos) requieren la utilización combinada de ambos esquemas para optimizar la capacidad de servicio del sistema.

Entre los dos esquemas de gestión, el esquema de mirroring es el que tiene una mayor preponderancia. La razón del mayor porcentaje de almacenamiento asignado al mirror estriba en la alta replicación asociada con el esquema de caching. Por lo tanto, solo es recomendable la utilización de la cache para almacenar contenidos con una alta popularidad (sobre todo cuando la capacidad de los servidores-proxy es escasa) que permitan incrementar el porcentaje de peticiones servidas localmente, y sin afectar a la eficiencia de los mirrors distribuidos.

La distribución de los contenidos en los mirrors de los servidores tiene una importante influencia sobre la distancia media de servicio de las arquitecturas P-Tree, y por lo tanto, afecta a la capacidad de servicio final del sistema LVoD.

En el capítulo se ha propuesto una heurística de asignación de videos a los mirrors cuyo objetivo es minimizar la distancia de servicio del sistema LVoD. Esta política de mapping, denominada ByDistance, ha demostrado su eficiencia en el incremento del rendimiento de la arquitectura Double P-Tree hasta un 65% respecto a políticas más sencillas.

El orden en el que se realizan las asignaciones de contenidos a cada uno de los mirrors influye, como hemos visto, en la eficiencia de las políticas de mapping. Por lo tanto, se han probado diferentes órdenes de procesamiento de los servidores (por niveles, Pre-orden y Post-orden) y se ha obtenido una diferencia de rendimiento entre ellos de un 15%. Los órdenes de procesamiento más eficientes son aquellos que permiten procesar consecutivamente los servidores-proxy que forman parte de un mismo mirror distribuido, para de este modo reducir los efectos colaterales entre las asignaciones realizadas. Los órdenes de procesamiento Post-Orden y por niveles son los que han demostrado los mejores resultados.

Capítulo 6

Políticas de balanceo del tráfico

Resumen

En este capítulo analizaremos la influencia sobre el rendimiento de las arquitecturas P-Tree del desbalanceo del tráfico en las redes locales. A partir del análisis realizado se propondrán dos clases distintas de políticas de balanceo del tráfico: políticas estáticas (mapping balanceado y ByDistance gradual) que se basan en equilibrar la popularidad de los contenidos asignados a los mirrors y una política dinámica de balanceo de peticiones que redistribuye las peticiones remotas de forma que se logre una distribución del tráfico más uniforme. Ambas políticas han demostrado una gran eficiencia en la reducción del desbalanceo del sistema.

6.1 Influencia de la distribución del tráfico en la eficiencia de las arquitecturas distribuidas

Un factor importante en la eficiencia de los sistemas distribuidos en general es el balanceo de la carga entre los distintos componentes. Si no se realiza un correcta distribución del trabajo entre cada uno de los componentes, puede ocurrir que mientras unos componentes del sistema están ociosos otros estén saturados. Éste desequilibrio entre los distintos componentes del sistema pueda afectar a su rendimiento final.

En los sistemas distribuidos de VoD existen dos tipos de recursos susceptibles de provocar desequilibrios: los servidores y las redes.

El desbalanceo en los servidores se puede producir por un desequilibrio en la distribución de las peticiones de los usuarios ó bien por la diferente conectividad de los servidores en la topología. En [Ven96] y [Ven97] se describen los tres principales mecanismos que se pueden utilizar para gestionar la carga de los servidores: replicación, reducción de la replicación y migración de las peticiones.

- *Replicación*: consiste en replicar contenidos más cargados en otros servidores de forma que se permita una redistribución de la carga.
- *Reducción de la replicación*: es el mecanismo inverso al anterior y consiste en liberar el almacenamiento asociado con las réplicas de contenidos multimedia infrautilizados.
- *Migración de peticiones*: Consiste en migrar peticiones existentes desde servidores saturados hacia otros servidores que no estén tan cargados. Esta técnica implica el restablecimiento de las conexiones de red, así como el intercambio de información de control de la petición entre los servidores implicados.

Estos autores proponen la utilización combinada de políticas para la asignación de los contenidos teniendo en cuenta su frecuencia de acceso (obtenida mediante métodos de predicción) y la planificación de las peticiones adaptadas a la carga del sistema.

El desbalanceo en las redes, por su parte, se puede generar a dos niveles: desbalanceo global entre el tráfico de las distintas redes ó servidores y desbalanceo local en el tráfico de los puertos de una misma red.

Las causas del desbalanceo a nivel global se pueden atribuir, de forma similar a los servidores, a la distinta conectividad de los nodos de servicio dentro de la topología ó bien al desequilibrio en la distribución de las peticiones de los usuarios.

En la figura 6-1 se muestran los diferentes volúmenes de carga de trabajo (definido en este contexto como el ancho de banda de servicio requerido para atender las peticiones activas) y tráfico que pueden soportar los distintos componentes de una arquitectura Double P-Tree. Los resultados se ha obtenido mediante la herramienta de simulación, cuando el sistema alcanza su máxima capacidad de servicio y utilizando los siguientes parámetros de configuración: 15 redes locales, 3 hermanos y 3 puertos de servicio. En dichos resultados podemos constatar el desequilibrio existente entre los distintos servidores y redes del sistema, el cual puede aumentar el porcentaje de peticiones denegadas en los servidores más saturados.

El desbalanceo global se debe a causas que no pueden ser controladas por el sistema (distribución de las peticiones de los usuarios) ó que vienen fijadas por la arquitectura (diferente conectividad). Por lo tanto, nosotros vamos a intentar analizar y solucionar los desequilibrios locales ya que pueden llegar a tener una mayor incidencia sobre el rendimiento de las arquitecturas distribuidas.

El desequilibrio local se produce cuando el tráfico de los distintos puertos no es homogéneo. Este desequilibrio puede llegar a tener una fuerte repercusión sobre los requisitos del sistema cuando se utilizan redes segmentadas. Como hemos visto en el capítulo 3 (apartado 3.2.2), los

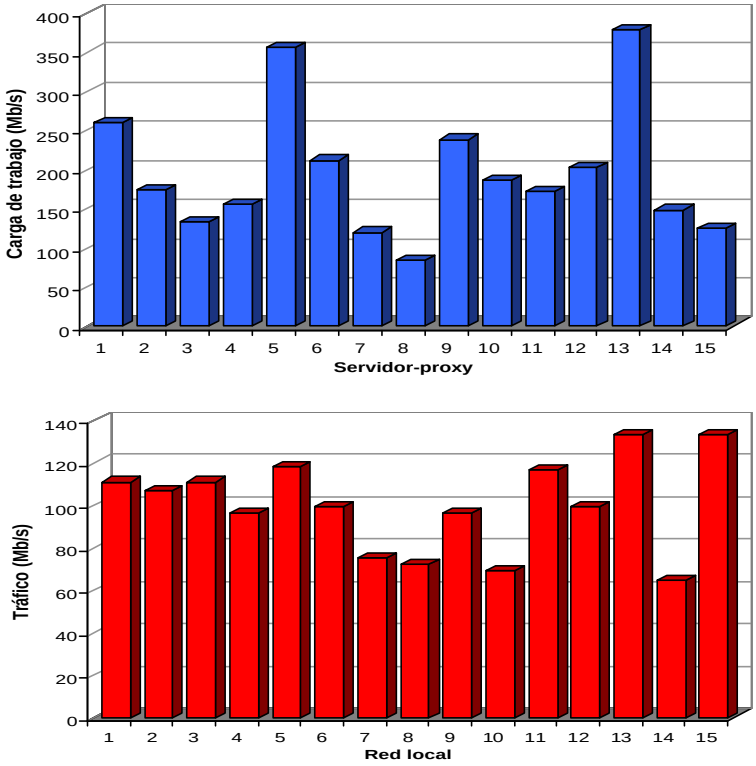


Figura 6-1. Distribución de la carga / tráfico del sistema entre los servidores-proxy y las redes locales de una arquitectura Double P-Tree

requisitos de una red segmentada dependen del puerto que recibe el máximo tráfico. Un desequilibrio en el tráfico de los puertos incrementará este máximo y con ellos los requisitos de ancho de banda de las redes del sistema.

Los resultados del capitulo 3 han demostrado que las redes segmentadas son ideales para los sistemas distribuidos como Proxy-Tree ó Double P-Tree. Si logramos que el tráfico de los puertos de las redes locales sea más homogéneo, estas arquitecturas pueden mejorar significativamente su rendimiento.

En este capítulo analizamos las causas del desequilibrio del tráfico en los puertos de las redes locales y proponemos diversas políticas para conseguir un tráfico más equilibrado que nos permita mejorar el rendimiento de las arquitecturas distribuidas de VoD.

6.1.1 Análisis del tráfico en los puertos de las redes locales

Tal y como hemos visto, los puertos de las redes locales en las arquitecturas P-Tree se pueden clasificar en 3 categorías: puertos de servicio (utilizados por el servidor-proxy para conectarse a la red), los puertos de la topología (utilizados en la conexión con las servidores hermanos, etc...) y por último los puertos de usuario (que conectan las redes de los usuarios a la red local).

En la figura 6-2 mostramos un ejemplo de la distribución del tráfico entre los distintos puertos de una red local en un sistema Double P-Tree. Los resultados expuestos se han obtenido mediante la simulación de una arquitectura Double P-Tree que utiliza redes locales segmentadas con un ancho de banda de 100 Mb/s y compuestas por 24 puertos. En la gráfica se muestran las

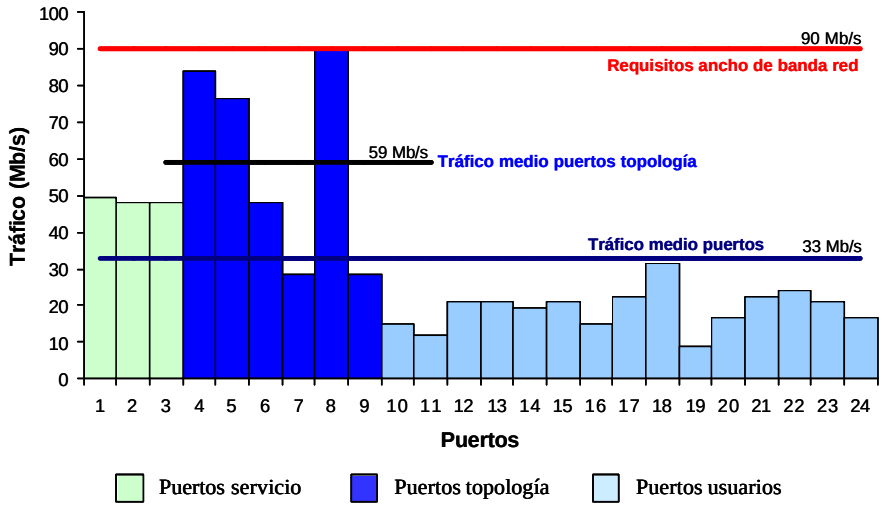


Figura 6-2. Distribución del tráfico en los puertos de la red en un sistema Double P-Tree

características del tráfico recibido por cada uno de los 3 tipos de puertos (servicio, topología, usuarios) en una red local cuando el sistema LVoD alcanza su máxima capacidad de servicio.

El tráfico de los puertos de servicio (puertos 1, 2 y 3) se muestran perfectamente equilibrados. Esta distribución no es una casualidad, sino que es inducida explícitamente por la política de gestión de los servidores locales. La distribución de este tráfico entre los puertos de servicio depende exclusivamente de las decisiones que se tomen en los servidores. Cualquier petición que llega a un servidor se puede servir indistintamente por cualquiera de sus puertos de servicio. Los servidores se pueden aprovechar de esta flexibilidad de elección de puerto de servicio para equilibrar su tráfico. Para lograr este objetivo, los servidores siempre escogen el puerto de servicio menos cargado para atender las nuevas peticiones que se produzcan en el sistema.

El tráfico de los puertos de usuario (puertos 10 a 24) tiene características diferentes a los puertos de servicio. El tráfico de estos puertos, como se puede ver en la figura, es bastante irregular. Este desequilibro es atribuible a la naturaleza no uniforme de las peticiones de los usuarios, las cuales se pueden distribuir de forma irregular entre los distintos puertos de usuario. El desbalanceo en los puertos de los usuarios es inevitable y poco se puede hacer para corregirlo. Además, la distribución del tráfico entre los puertos de los usuarios no suele influir en los requisitos de la red debido a que al disponer de una mayor cantidad de puertos, el tráfico máximo que recae en cada uno de ellos suele estar por debajo del tráfico medio de la red (por lo tanto, no suelen ser los culpables del puerto con el tráfico máximo).

El máximo responsable del incremento de los requisitos de red lo podemos encontrar en los puertos de la topología (puertos 4 a 9). Se puede apreciar a primera vista, que el tráfico de estos puertos está más descompensado ya que entre el puerto más saturado (90 Mb/s) y el menos cargado (30 Mb/s) hay un diferencia de tráfico superior a los 61 Mb/s.

Este desequilibrio provoca un aumento de los requisitos de la red hasta los 90 Mb/s. Si el tráfico en los puertos estuviese más balanceado, se podría alcanzar una reducción máxima del ancho de banda de la red de un 33% hasta los 59 Mb/s.

6.1.2 Causas del desequilibrio del tráfico en los puertos de la topología

Para balancear el tráfico asociado con los puertos de la topología, primero necesitamos conocer las razones que provocan dicho desequilibrio. Existen dos causas: la diferencia de popularidad de los contenidos de los mirrors y la conectividad de los nodos en la topología.

El tráfico que circula por los puertos de la topología es generado exclusivamente por las peticiones remotas, que son las únicas que utilizan estos puertos. El puerto concreto por el cual

se va a transmitir una petición depende del mirror remoto en donde se encuentra el contenido requerido para servir la petición remota.

El tráfico generado hacia un servidor remoto es proporcional a la popularidad de los contenidos almacenados en su mirror. Si uno de los mirrors tiene asignados contenidos con una frecuencia de acceso superior a los videos asignados al resto de mirrors vecinos, recibirá un mayor volumen de peticiones remotas, provocando un desbalanceo en el tráfico de los puertos de la topología. Por lo tanto, la popularidad de los contenidos en los mirrors es una de las fuentes del desequilibrio de tráfico en los puertos de la topología.

Ahora bien, el tráfico asociado con las peticiones remotas se puede dividir en dos tipos: el tráfico remoto de entrada (compuesto por peticiones remotas atendidas por el servidor local) y el tráfico remoto de salida (compuesto por las peticiones que no pueden ser atendidas localmente).

El tráfico remoto de entrada depende principalmente de la popularidad de los contenidos almacenados en el mirror local. Esta popularidad es la misma para todos los puertos de la topología de la red local, por lo tanto el volumen de tráfico recibido por cada uno de ellos será muy similar. Por ejemplo, un servidor-proxy cuyos contenidos del mirror tengan una popularidad del 11%, implica que cada unos de los servidores adyacentes le redirigirán el 11% de sus peticiones locales. Por lo tanto, en promedio cada uno de los puertos de la topología recibirá un tráfico de entrada alrededor del 11% del tráfico local de los servidores remotos.

La distribución del tráfico remoto de salida entre los distintos puertos de la topología depende de la popularidad de los contenidos almacenados en los mirrors de los servidores-proxy remotos. Si la popularidad de los contenidos de un mirror remoto es considerablemente mayor que el resto de mirrors adyacentes, esto implica que el tráfico de salida asociado con el puerto de la topología que conecta con este servidor también será considerablemente mayor, provocando un desbalanceo.

Podemos concluir que el desbalanceo del tráfico asociado con los mirrors se produce únicamente con el tráfico remoto de salida y, por lo tanto, si queremos evitar este desequilibrio debemos trabajar con las peticiones que no se pueden atender localmente en los servidores locales.

La conectividad de la topología puede ser también una fuente de desequilibrio cuando uno de los puertos de la topología recibe un mayor volumen de tráfico debido a la menor capacidad de un mirror distribuido.

Un ejemplo de este tipo de desbalanceo se puede localizar en el nodo superior de la topología Double P-Tree. Este nodo únicamente tiene 2 servidores vecinos (los nodos hijos) y por lo tanto, todas las peticiones que no se puedan servir localmente tendrán que cruzar estos dos servidores.

Este alto volumen de tráfico recibido en los hijos desde el padre (servidor remoto del nivel superior) puede generar un desbalanceo en los puertos de la topología. El mismo efecto se produce en los nodos de los últimos niveles de la topología. El desbalanceo generado por la conectividad de la topología se puede generar tanto por el tráfico de entrada como el tráfico de salida.

Para poder exponer de forma más clara estas dos fuentes de desbalanceo vamos ha utilizar un ejemplo. La figura 6-3 muestra la asignación de videos realizada mediante la política de asignación ByDistance (descrita en el apartado 5.3.2 del capítulo 5) en un sistema Double P-Tree compuesto por 15 servidores. El número situado debajo de los contenidos asignados al mirror de cada uno de los servidores-proxy representa la popularidad total de los contenidos asignados a ese mirror, es decir, la popularidad del 46% en el mirror del servidor-proxy número 1 implica que el 46% de las peticiones del sistema acceden a estos contenidos.

Si nos fijamos en el servidor número 2 y en la popularidad de los mirror de los nodos vecinos, podemos darnos cuenta de los desequilibrios generados. Mientras que los contenidos de los mirrors de los servidores remotos 3, 4 y 5 tienen una popularidad total del 14%, 11% y 9% respectivamente, el mirror de servidor 1 tiene una popularidad mucho mayor, alcanzando el 46%. Esta asignación de los contenidos significa que en una primera aproximación (sin tener en cuenta otros factores) el 46% del tráfico de este servidor utilizará el puerto de conexión con el servidor 1, mientras que el resto de puertos de la topología recibirán únicamente el 14%, 11% y 9% del volumen de tráfico. Esta distribución del tráfico muestra como un desequilibrio en la popularidad de los contenidos de los mirrors puede generar un desbalanceo en los puertos de las redes

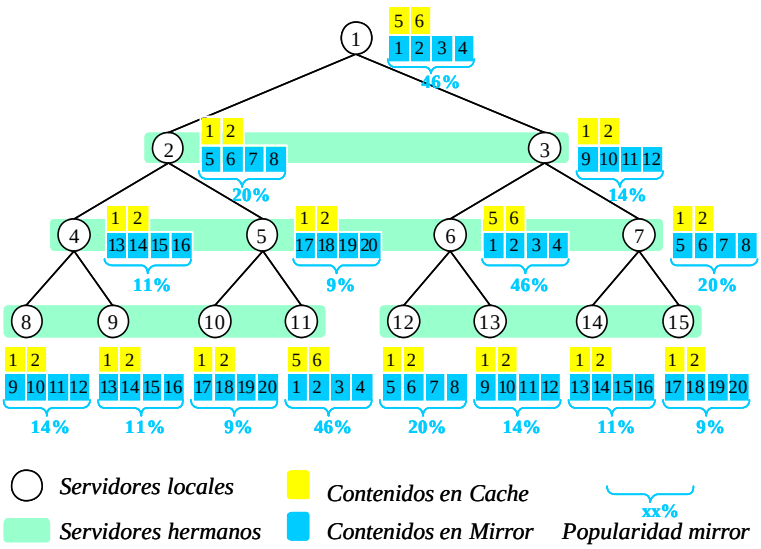


Figura 6-3. Desequilibrio de la popularidad en los mirrors

locales.

A continuación analizaremos los desequilibrios generados por la topología, para ello nos vamos a fijar en el tráfico asociado a las peticiones remotas recibidas por el servidor 2 y en la conectividad de los nodos vecinos. El tráfico remoto del servidor 4 por ejemplo, se tiene que distribuir entre los 6 servidores adyacentes (2, 5, 6, 7, 8 y 9), el tráfico remoto del servidor 2 se distribuye entre 4 servidores (1, 3, 4 y 5) y el tráfico del servidor 1 entre únicamente 2 servidores adyacentes (2 y 3). Esto significa que el servidor número 2 recibirá una sexta parte de tráfico remoto de los servidores 4 y 5, una cuarta parte del tráfico remoto del servidor 3 y la mitad del tráfico remoto del servidor 2. Esta distribución del tráfico vuelve a generar un desequilibrio en los puertos de la topología de la red local de este servidor. El puerto más saturado vuelve a ser el de conexión con el primer servidor, lo cual aumenta el desequilibrio ya existente debido a la popularidad de los contenidos de los mirrors.

6.2 Políticas de balanceo del tráfico en los puertos

En el apartado anterior hemos mostrado como el desbalanceo del tráfico en los puertos de las redes locales puede incrementar los requisitos de ancho de banda de las redes del sistema y las causas que provocan estos desequilibrios. En esta sección, vamos a proponer una serie de políticas para reducir estos desequilibrios. Estas políticas se pueden clasificar en dos grupos: estáticas y dinámicas, en función de si las decisiones de balanceo se toman durante la etapa de inicialización del sistema (estáticas) ó a medida que se reciben las peticiones de los usuarios durante el funcionamiento del sistema (dinámicas) [Cor03] [Cor03b].

Las políticas estáticas tratan de equilibrar el tráfico de las redes, asignando los videos a los mirrors de forma más equilibrada, es decir, de forma que la popularidad de los contenidos en los distintos servidores sea parecida. Estas políticas solo se llevan acabo durante la etapa de asignación de los contenidos y se mantiene sin cambios durante el funcionamiento normal del sistema.

Por el contrario, las políticas dinámicas balancean el tráfico a medida que éste se va generando y utilizando para ello la información disponible en cada instante. Estas políticas permiten adaptarse mejor a los cambios que se puedan producir en el volumen y tipo de tráfico generado en el sistema.

Para evaluar las políticas de balanceo del tráfico vamos a calcular el desbalanceo en el tráfico (D) en un conjunto de puertos (pi-pf) de una red local mediante la siguiente expresión:

$$D_{pi,pf} = \frac{TráficoMáximo(pi, pf) - TráficoMínimo(pi, pf)}{TráficoMedio(pi, pf)} \quad (6.1)$$

Esta formula nos va permitir evaluar el desbalanceo existente en las redes locales de sistema LVoD, teniendo en cuenta el desequilibrio existente en todos los puertos de la red (desbalanceo global) y el desbalanceo teniendo sólo en cuenta los puertos de topología de la red.

Mediante la expresión (6.1) y a partir de las estadísticas obtenidas mediante la simulación de una arquitectura Double P-Tree con 63 redes locales, hemos evaluado la efectividad de las distintas políticas de balanceo de tráfico presentadas en este capítulo. Para evaluar el desbalanceo se calcula el desequilibrio global y de la topología promedio de todas las redes locales que componen la arquitectura Double P-Tree evaluada.

6.2.1 Políticas estáticas de balanceo del tráfico

El primer método que podemos utilizar para solucionar el desbalanceo se basa en modificar las políticas de mapping (vistas en el capítulo 5) para que generen distribuciones más equilibradas respecto a la popularidad de los contenidos almacenados en cada uno de los servidores del sistema.

Hemos propuesto dos políticas de mapping diferentes cuyo objetivo principal es reducir el desbalanceo del tráfico: la política de asignación balanceada de contenidos y la política de asignación de videos de forma gradual. Ambas políticas de asignación son una modificación del mapping ByDistance, que como ya hemos visto en el capítulo 5, es la política de asignación que mejor resultados consigue respecto a la distancia de servicio del sistema.

En todas estas políticas de mapping, se asume que se dispone de información de la frecuencia de acceso de los contenidos. Esta información se puede obtener mediante una estimación previa (como la Zipf utilizada a lo largo de toda esta tesis) ó mediante estadísticas históricas. Al disponer de esta información nos permite ordenar los contenidos en función de su popularidad (el primer video es el más popular y el último es el menos popular). Las distintas políticas de mapping tendrán en cuenta esta información para alcanzar un mejor equilibrio entre la popularidad de los contenidos almacenados en los distintos servidores locales.

6.2.1.1 Mapping balanceado

Esta política de asignación se basa en equilibrar los videos más populares con los videos con menos popularidad. De esta forma, cuando hay más de un video (*n*) a la máxima distancia, asignaremos por parejas el n-esimo video más popular y el n-esimo video con menor

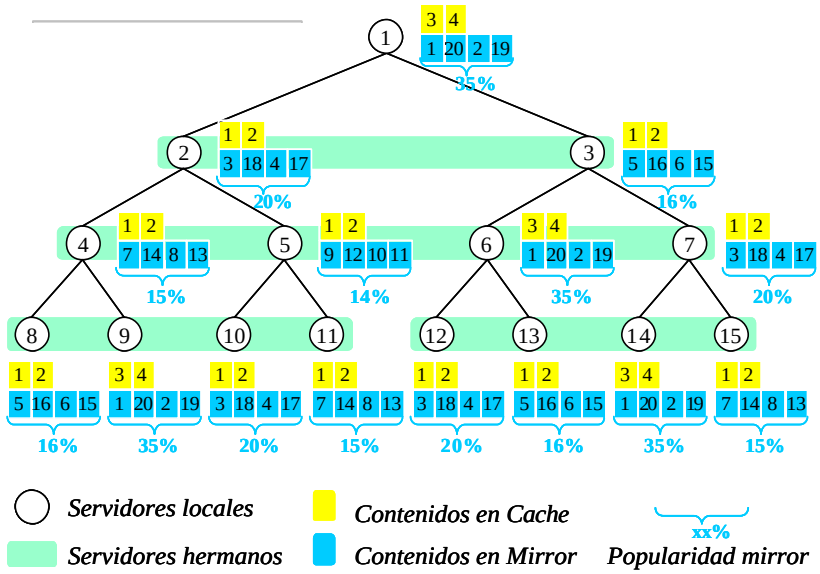


Figura 6-4. Asignación de videos utilizando la política de mapping balanceado

popularidad, logrando una frecuencia de acceso más equilibrada entre los distintos servidores locales.

En la figura 6-4, mostramos el mapping obtenido con esta asignación y la popularidad total los contenidos de los servidores. La asignación resultante se ha obtenido después de aplicar el mapping balanceado para asignar los contenidos a los mirrors de una arquitectura Double P-Tree con 15 redes locales y 3 hermanos. Una vez asignados los contenidos de los mirrors se ha inicializado la cache de cada uno de los servidores (para poder así calcular la distancia media de servicio) con los videos más populares que no estén ya asignados al mirror local.

Si nos fijamos en la popularidad total de los contenidos asignados a los mirrors podemos ver que con la nueva política de asignación la popularidad de los mirrors está más equilibrada que cuando utilizamos la política ByDistance. Si en la figura 6-3, la popularidad máxima en los mirror s era del 46% (cuando se asignan los 4 videos más populares a un mismo mirror) con la asignación de videos balanceados se ha conseguido reducirla hasta el 35%. Lo mismo ocurre con la popularidad mínima que pasa de un 9% (cuando se asignan los 4 videos menos populares a un mismo mirror) hasta el 14%. La reducción de los extremos de popularidad permite reducir el desbalanceo entre la frecuencia de acceso de los mirrors desde 37% al 21%.

La tabla 6-1 muestra la distancia media de servicio y el desbalanceo del tráfico en las redes del sistema cuando se utilizan las políticas de asignación de videos secuencial, ByDistance y balanceado.

Tal y como podemos ver, el mapping balanceado permite reducir el desbalanceo global un 20% y el desbalanceo en los puertos de la topología en un 30%, respecto a la política secuencial. Con respecto a la política ByDistance los resultados no son tan buenos, tal y como se muestra en la tabla.

Tabla 6-1. Desbalanceo de tráfico asociado con las políticas de asignación de videos Secuencial, ByDistance y Balanceado

Política de asignación de videos	Distancia media de servicio	Desequilibrio del tráfico en los puertos	
		Global	Topología
Secuencial	1,780	54,46%	26,62%
ByDistance	1,747	43,60%	19,69%
Balanceado	1,769	43,53%	18,05%

Sin embargo, los beneficios derivados de la reducción de requisitos de red obtenida por esta política se anulan parcialmente debido al aumento de la distancia media de servicio del sistema desde 1,747 (mapping ByDistance) hasta 1,769 (mapping balanceado) y el consecuente incremento del ancho de banda de red requerido. Por lo tanto la política de balanceo estático no solo tiene que equilibrar la frecuencia de acceso de los contenidos, si no que también tiene que minimizar la distancia de servicio si no quiere reducir el rendimiento del sistema de VoD.

6.2.1.2 Política de asignación de videos gradual

Tal y como hemos visto en la tabla 6-1, tanto las políticas de asignación secuencial y ByDistance generan desequilibrios de tráfico en los puertos de las redes locales. Este defecto se debe en parte a que todos los videos de un mirror son asignados de golpe. Esto unido a la tendencia natural de la política a escoger los videos más populares para minimizar la distancia de servicio, provoca un desequilibrio entre la popularidad de los contenidos asignados a los distintos servidores.

Para reducir este desequilibrio, pero al mismo tiempo mantener una distancia de servicio pequeña, proponemos la realización de una asignación de los videos de forma gradual en los mirrors.

Para conseguir un mapping más progresivo vamos a modificar la política ByDistance de forma que la asignación de los videos se realice paso a paso. Primero se asignará una determinada cantidad k de videos en cada uno de los servidores. Este procedimiento se repetirá mientras quede almacenamiento disponible en alguno de los mirrors. Normalmente el valor de K utilizará

valores pequeños (1 ó 2) para implementar una asignación más gradual de los videos. En el caso de que K sea igual a la capacidad de videos del mirror, entonces esta política se comporta igual que la política ByDistance original.

Además de equilibrar los contenidos de los mirrors, otra ventaja de las asignaciones graduales estriba en que permite que los mirrors de los servidores se adapten a los contenidos que han sido previamente asignados en los servidores vecinos.

El algoritmo resultante de esta optimización lo mostramos en la figura 6-5. En el nuevo algoritmo, se añade un nuevo bucle iterativo (respecto al presentado en el capítulo 5, apartado 5.3.2) mediante el cual se intentará realizar un nuevo paso de mapping mientras haya algún mirror que no este totalmente lleno. En cada uno de los pasos se intenta asignar K videos a todos los servidores del sistema que no estén llenos. Al final de procesar cada uno de los servidores se comprueba si el servidor todavía tiene espacio en el mirror para algún video adicional, si es así se habilita la siguiente iteración del algoritmo.

En la figura 6-6, mostramos mediante el mismo ejemplo anterior la asignación obtenida mediante la política ByDistance Gradual. Como podemos ver la frecuencia de acceso total de los videos en los mirrors es más equilibrada. Con esta distribución conseguimos reducir la popularidad máxima de los mirrors al 33% y aumentamos la popularidad mínima hasta el 15%. El

```
Do
    TodosMirrosLlenos  = Cierto
    Foreach ServidorLocal
        If not ServidorLocal.MirrorLleno() then
            CalcularDistanciaContenidos(ServidorLocal)
            For i=0 To K-1
                videoId = ObtenerContenidoMásAlejado()
                ServidorLocal.SetContenidoMirror(videoId)
            EndFor
        Endif
        If not ServidorLocal.MirrorLleno() then
            TodosMirrosLlenos  = Falso
        Endif
    EndForeach
While TodosMirrosLlenos
```

Figura 6-5. Pseudo-código de la política de mapping ByDistance Gradual

desequilibrio máximo entre los dos mirrors queda reducido hasta el 18%.

Esta asignación cumple el objetivo de reducir el desequilibrio del tráfico de las redes locales. Los resultados de la tabla 6-2 revelan que la política de asignación de contenidos graduales consiguen obtener una distribución del tráfico más balanceado.

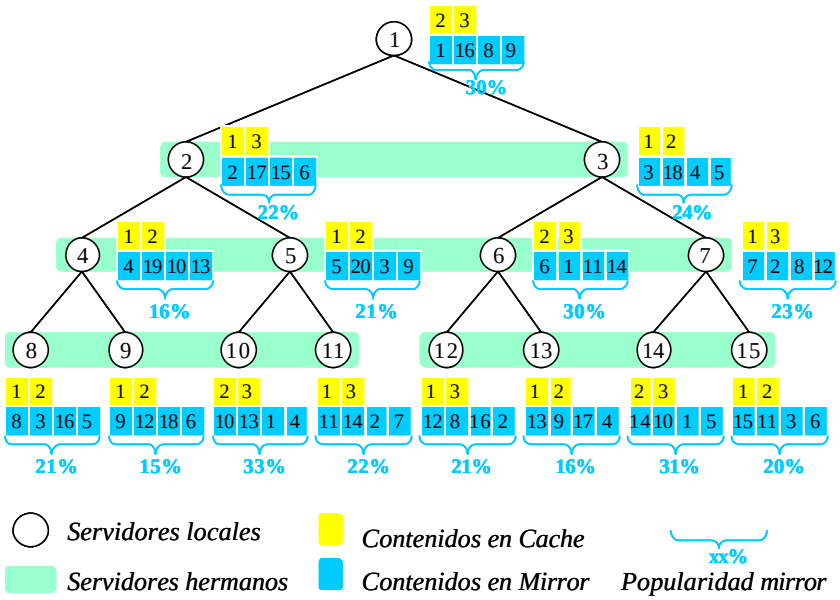
Tabla 6-2. Desbalanceo del tráfico asociado con la política de asignación ByDistance gradual

Política de asignación de videos	Distancia media de servicio	Desequilibrio del tráfico en los puertos	
		Global	Topología
ByDistance	1,747	43,60%	19,69%
ByDistance Gradual	1,742	31,00%	13,28%

La política de asignación ByDistance Gradual mejora claramente los resultados de la política

Figura 6-6. Asignación de videos obtenida con la política ByDistance Gradual

ByDistance original. El desbalanceo global en las redes locales se reduce en un 29% (pasando del 43,6% al 31%) y el desbalanceo de los puertos de topología en un 32% (13,28% en contraposición a 19,69%). Estos resultados se logran manteniendo la distancia de servicio en valores muy parecidos a la política original y por lo tanto, sin aumentar el ancho de banda requerido por las peticiones.



Si comparamos estos resultados con los obtenidos previamente por la política de mapping balanceado (Tabla 6.1) se pone de manifiesto el mejor rendimiento de las políticas de asignación gradual. La política ByDistance gradual consigue un tráfico más equilibrado, tanto si tenemos en cuenta el desequilibrio global (31,00 frente al 43,53%) ó el desequilibro en los puertos de la topología (13,28% frente al 18,05%). Además, el mapping ByDistance gradual no solo no incrementa la distancia media de servicio sino que consigue una ligera reducción de la misma (1,742 respecto a 1,747 con el mapping ByDistance original).

Sin embargo, estas técnicas no logran reducir el desbalanceo provocado por otras causas, como por ejemplo, la conectividad de la topología. Además, debido a su naturaleza estática, estas técnicas no se adaptan bien variaciones producidas en las preferencias de los usuarios, que pueden provocar un cambio en la popularidad de los contenidos.

Para subsanar estas carencias proponemos una política de balanceo dinámica con el objetivo de lograr una mejor distribución de las peticiones remotas entre los servidores adyacentes. Esta política se denomina Balanceo de peticiones y se describe en el siguiente apartado.

6.2.2 Política de balanceo dinámica de peticiones

Esta política de balanceo de peticiones se basa en la selección de los servidores remotos que deben gestionar las peticiones que no se pueden servir localmente. Estas peticiones son las que generan la mayor parte del tráfico que circula por los puertos de la topología y por lo tanto, las que provocan el desequilibro de estos puertos.

Cuando una petición no se puede servir localmente, el servidor local busca gestores alternativos entre los servidores adyacentes. Si alguno de los servidores candidatos contesta que dispone de recursos suficientes para gestionar la petición, el servidor local redirige la petición hacia este servidor, el cual se encarga de servirla al usuario.

La política de balanceo de peticiones entra en juego en el caso de que hayan diversos servidores candidatos a una misma distancia para atender una petición remota. De esta forma, siempre se otorga más prioridad al criterio de reducción de la distancia de servicio sobre el criterio de balanceo del tráfico, con el objeto de mantener la eficiencia del sistema.

La política de balanceo de peticiones seleccionará siempre el candidato cuyo camino entre el servidor remoto y el usuario esté menos saturado. De esta forma, se evita la aparición de enlaces sobresaturados y se logra un tráfico en las redes locales más equilibrado. Si sólo hay un servidor alternativo a la mínima distancia posible, éste será el encargado de atender la petición y no se activará la política de balanceo.

La selección del servidor candidato se realiza sólo teniendo en cuenta el equilibrio del tráfico de los puertos de la topología del servidor local antes que seleccionar la ruta completa menos saturada. Este criterio nos permite el balanceo de las peticiones únicamente utilizando información sobre el tráfico de los puertos de la topología locales (no se necesita disponer de información sobre el tráfico de todas las redes que debe cruzar el stream de servicio de la petición). Estas estadísticas se pueden obtener a partir del número de peticiones que se están actualmente sirviendo por cada uno de los puertos de la topología del servidor local.

En la figura 6-7, se puede ver mediante un ejemplo el funcionamiento de la política de balanceo de peticiones. Se asume que el servidor número 4 recibe una petición para el video número 3, desde un usuario local.

Inicialmente se comprueba si la petición se puede gestionar localmente; éste no es el caso ya que el video requerido no se encuentra entre los contenidos del servidor. A continuación se intenta localizar un servidor alternativo entre los servidores adyacentes que pueda atender la petición. Los servidores 2, 5, 6, 7 y 8 pueden gestionar la petición remota ya que disponen del contenido requerido. Como hay más de un candidato posible, se activa el algoritmo de balanceo de peticiones para seleccionar cual de los candidatos permite obtener un tráfico más equilibrado en la red local. La política evalúa el tráfico de los puertos de la topología asociado con cada uno de los servidores alternativos (para el ejemplo asumiremos un tráfico acumulado en cada uno de los puertos de la topología de 60, 30, 50, 20 y 40 Mb/s respectivamente). Finalmente, se selecciona el servidor 7 para atender la petición remota ya que el puerto que tiene que utilizar es el que está menos cargado (el puerto que se utiliza en la conexión del servidor 4 con el servidor hermano 7 únicamente soporta un tráfico acumulado de 20 Mb/s).

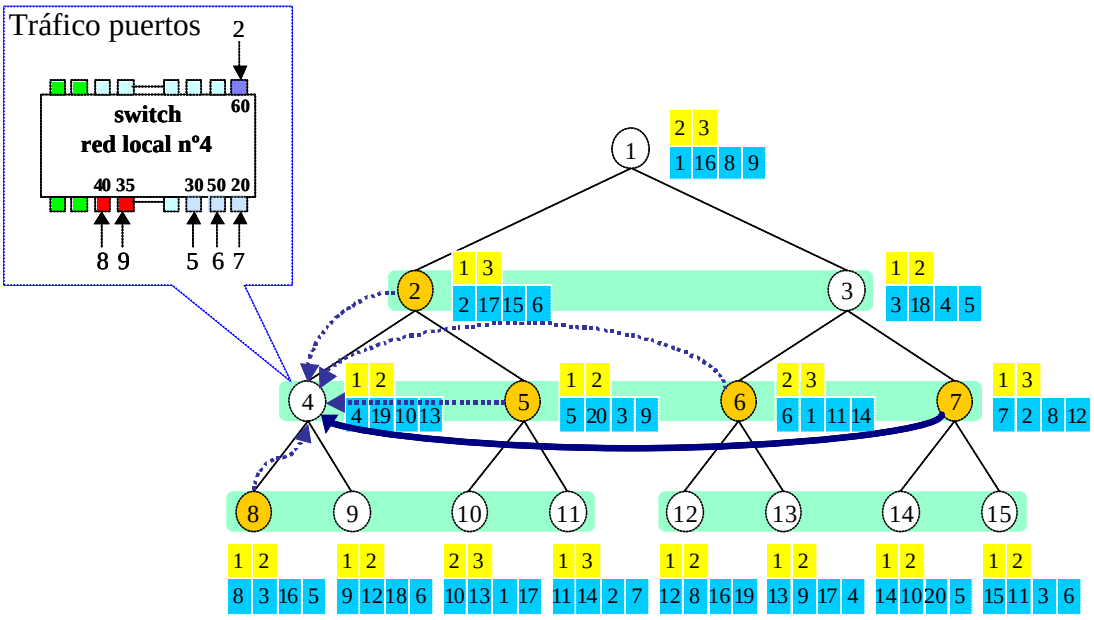


Figura 6-7. Ejemplo del funcionamiento de la política de balanceo de peticiones

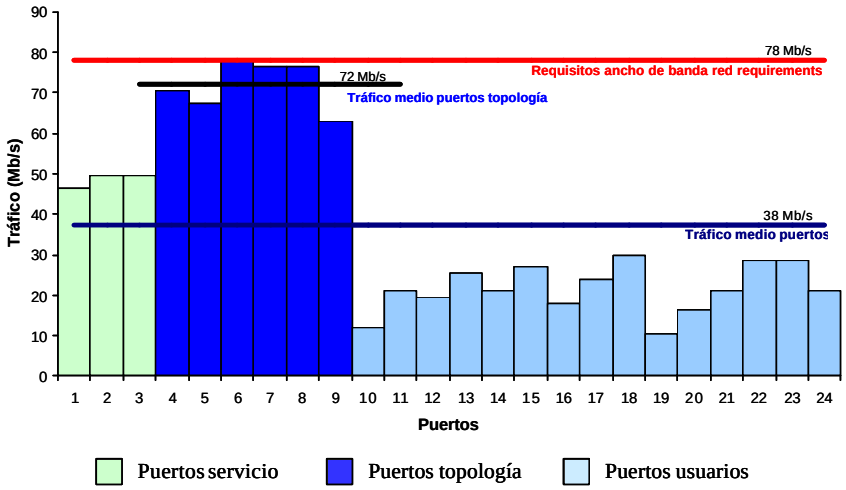


Figura 6-8. Influencia de la política de balanceo dinámica de peticiones sobre el tráfico de un sistema Double P-Tree

Para evaluar la eficiencia de esta política, en la figura 6-8 mostramos como se ve afectado el tráfico en la red de la figura 6-2 cuando se aplica la política de balanceo de peticiones. Podemos constatar que el tráfico resultante en los puertos de la topología está mucho más balanceado. Ahora la diferencia entre el pico máximo y el mínimo apenas supera los 14.000 Mb/s (78.000-64.000 Mb/s) frente a los 60.000 Mb/s que se alcanzan cuando no se controla el balanceo del tráfico.

También podemos ver que a pesar de que el volumen de tráfico de la red se ha incrementado un 25% (desde 32.875 a 37.563 Mb/s), los requisitos de ancho de banda de la red se han reducido en un 15% hasta los 78.000 Mb/s

En la tabla 6-3 muestra una comparativa entre el desbalanceo obtenido en el sistema con la utilización ó no de la política de balanceo de peticiones. En la tabla se compara la mejor política de balanceo estática (ByDistance gradual) con las políticas de balanceo dinámicas (balanceo de peticiones) y una combinación híbrida de ambas políticas (ByDistance gradual más balanceo de peticiones).

Tabla 6-3. Desbalanceo del tráfico asociado con la política de balanceo de peticiones

Política de balanceo de tráfico	Distancia media de servicio	Desequilibrio del tráfico en los puertos	
		Global	Topología
ByDistance Gradual	1,742	31,00%	13,28%
Balanceo de peticiones	1,747	31,39%	13.58%
Balanceo de peticiones Gradual	1,742	17,79%	7,10%

Podemos observar que las políticas de balanceo de peticiones consiguen reducir significativamente el desequilibrio del tráfico en las redes del sistema, obteniendo un desequilibrio global del 31% y un desequilibrio en los puertos de la topología del 13%. Si comparamos los resultados de esta política con los obtenidos cuando no se utiliza una política de balanceo de tráfico (ByDistance, en tabla 6.1) se constata una reducción del desbalanceo en lo puertos de la topología y global, logrando resultados similares a los obtenidos por la mejor de las políticas estáticas (ByDistance Gradual).

Una conclusión importante que podemos obtener del análisis de estos resultados es la total compatibilidad existentes entre las políticas estáticas (ByDistance gradual) y las políticas dinámicas (balanceo de peticiones). La utilización combinada de ambas políticas permite obtener los mejores resultados en el balanceo de tráfico en las redes del sistema, reduciendo el desequilibrio global hasta el 18% y el desequilibrio en los puertos de la topología hasta el 7%. La mejoras obtenidas mediante la política de balanceo de peticiones no afectan a la distancia media de servicio ya que la política no modifica las asignaciones de videos originales (ByDistance ó ByDistance gradual).

6.3 Optimización de la eficiencia de las políticas de balanceo de peticiones

Tal y como hemos visto, la política de balanceo de peticiones logra reducir considerablemente el desbalanceo de la carga de los puertos de la topología. Sin embargo, aún a pesar de su efectividad todavía se mantiene un desbalanceo superior al 7%.

La principal causa que no permite que la política alcance su máxima eficiencia estriba en el limitado número de veces que la política puede actuar. Para poder evaluar este problema, en la tabla 6-4 mostramos el porcentaje de peticiones gestionadas local y remotamente, y cuantas de estas últimas permiten la activación de la política de balanceo. Estos resultados se han obtenido a partir de las estadísticas generadas por la simulación de una arquitectura Double P-Tree con 63 redes locales y una capacidad en los servidores del 20% y utilizando la política de balanceo de peticiones.

Podemos observar que en el sistema utilizado como ejemplo, el 53% de las peticiones se gestionan localmente y el alrededor del 47% en servidores remotos. Del total de peticiones remotas, el 30% (el 14% del total) permiten la activación de la política de balanceo ya que se benefician de más de un servidor remoto candidato.

Tabla 6-4. Distribución de las peticiones en un sistema Double P-Tree

Peticiones					
Locales	**% Total**	**Remotas**	**% Total**	**Remotas susceptibles de ser balanceadas**	**% Total**
22.534	53,11%	19.893	46,89%	6.122	14,43%

La insuficiente capacidad de maniobra de la política de balanceo de peticiones se debe a la poca replicación de contenidos existentes en el mirror distribuido a distancia 2, lo cual reduce la probabilidad de que haya más de un candidato para servir una petición remota. La escasa replicación en los mirrors distribuidos se debe atribuir a las políticas de asignación de videos, que justamente procuran reducir el número de contenidos replicados en el mirror distribuido para minimizar la distancia de servicio del sistema.

Por lo tanto, podemos deducir que para incrementar la eficiencia de la política de balanceo de peticiones debemos, de alguna forma, aumentar la replicación de contenidos y con ello la capacidad de activación de la política dinámica. El incremento de la replicación se debe realizar con sumo cuidado para que no afecte a la distancia de servicio del sistema. Un aumento excesivo de la distancia puede llegar a cancelar todos los beneficios derivados de la política de balanceo y reducir la eficiencia.

De las posibles alternativas que podemos abordar para aumentar la replicación, tenemos que descartar las basadas en la cache autónoma. La replicación derivada del uso del esquema de caching puro no interesa ya que estos contenidos no suelen ser utilizados para gestionar peticiones remotas. Por lo tanto, la replicación asociada con este esquema de gestión no permite la intervención de la política de balanceo de peticiones.

Otra alternativa más viable consiste en incrementar la capacidad de los mirrors en los servidores locales. Este método consigue aumentar el tamaño del mirror distribuido y con ello, la probabilidad de que aparezcan contenidos replicados. Sin embargo existen dos motivos que nos inducen a creer que esta alternativa no es muy conveniente. Primero, al incrementar la capacidad del mirror, se reduce la eficiencia de la cache y con ello el rendimiento del sistema. Segundo, el aumento de la replicación de los contenidos de los mirrors afecta a todos los videos por igual, independientemente de su popularidad y frecuencia de acceso. Este característica no nos interesa ya que la efectividad de los videos replicados dentro de la política de balanceo es proporcional a su frecuencia de acceso. Cuanto más popular sea un contenido mayor será el volumen de tráfico que la política podrá gestionar para reducir el desbalanceo de los puertos de la red.

Por lo tanto, para mejorar la maniobrabilidad de la política de balanceo, utilizando para ello pocos recursos de almacenamiento, no interesa una replicación indiscriminada de contenidos poco accedidos, sino una replicación limitada (que no perjudique demasiado a la distancia de servicio) de contenidos populares. Para cumplir estos requisitos proponemos la utilización de un tercer esquema de gestión de los contenidos en los servidores locales: el caching cooperativo.

6.3.1 Caching distribuido cooperativo

Mediante el caching cooperativo, un conjunto de servidores gestionan de forma conjunta una cierta capacidad de almacenamiento para guardar los contenidos más populares.

El caching cooperativo se puede ver como una política intermedia entre los esquemas de gestión de caching y mirroring. La principal diferencia entre los tres esquemas de gestión estriba en el número de veces que cada contenido aparece replicado dentro del almacenamiento distribuido.

En el caso de que cada contenido aparezca una única vez dentro del almacenamiento distribuido estaríamos hablando de que se comporta como un mirror distribuido. Sin embargo, si los contenidos más populares aparecen replicados en todos los servidores que componen el almacenamiento distribuido, entonces se trata de un esquema de caching autónomo puro.

Ahora bien, en el caso de que los contenidos aparezcan R veces replicados (siendo R mayor que 1 y menor que el número de servidores que componen el almacenamiento distribuido) se trata de una cache cooperativa.

Mediante la inclusión de la cache cooperativa, el almacenamiento de los servidores-proxy queda distribuido entre 3 esquemas de gestión diferentes (tal y como muestra la figura 6-9): caching autónomo, caching cooperativo y mirroring.

En las arquitecturas P-Tree, la cache cooperativa esta formada por todos los servidores adyacentes, los cuales gestionan los contenidos de esta cache de forma conjunta. La capacidad de la cache cooperativa se puede evaluar de forma similar al tamaño de los mirrors distribuidos, multiplicando el número de servidores locales situados a distancia 2 de la red local por la capacidad del servidor asignada a este esquema de gestión. Al agrupar los servidores adyacentes podemos lograr una capacidad global considerablemente mayor con un pequeño porcentaje del almacenamiento de los servidores locales.

Figura 6-9. Esquemas de gestión de los contenidos en los servidores-proxy

El objetivo de la cache cooperativa no consiste en reducir la distancia de servicio (mediante la cache cooperativa se incrementa levemente la distancia de servicio), sino en mejorar la eficiencia de la política de balanceo de peticiones. Además, este tipo de caching distribuido, al contrario de lo que ocurre con la cache autónoma, se puede utilizar para gestionar peticiones remotas debido a que sólo un número reducido de los servidores adyacentes disponen de los mismos contenidos en la cache cooperativa.

Funcionamiento de la cache distribuida cooperativa

El funcionamiento de esta cache cooperativa es bastante similar al de la cache autónoma. En la figura 6-10, mostramos el pseudo-código del algoritmo de gestión correspondiente.

El algoritmo de gestión de la cache se activa cuando se produce un fallo al servir una petición local (debido a que el contenido requerido no está disponible) y una vez seleccionado el servidor remoto que debe gestionar la petición. Primero se activa la política de gestión de la cache de primer nivel (cache autónoma) y si el contenido candidato no tiene suficiente popularidad para ser almacenado en el primer nivel de cache, entonces se activa la política de gestión para la cache cooperativa.

Cuando se activa esta política, se comprueba si existen suficientes recursos de ancho de banda de red para poder recibir el contenido candidato desde el servidor remoto. En el caso negativo, el contenido la cache cooperativa no se modifica.

Figura 6-10. Pseudo-código de la política de gestión de la cache cooperativa

En el caso afirmativo, se evalúa si el contenido candidato cumple con los requisitos necesarios para poder ser almacenado en la cache cooperativa. Se han de cumplir dos requisitos:

1. La frecuencia de acceso del contenido candidato debe estar entre los candidatos más populares sin tener en cuenta los contenidos almacenados en el primer nivel de cache.
2. El número de replicas del video candidato existentes en la cache cooperativa no supere el máximo permitido (R, factor de replicación de contenidos en la cache cooperativa).

Esta condiciones se evalúan contra una tabla que contiene el número de accesos y el número de replicas existentes para cada video del catálogo del sistema. Las estadísticas de esta tabla se calculan teniendo en cuenta los contenidos almacenados en todos los servidores locales que componen la cache cooperativa.

Si estos dos requisitos se cumplen, se debe comprobar además que existe suficiente capacidad de almacenamiento disponible en la cache cooperativa para guardar el contenido candidato. En el caso que no haya suficiente almacenamiento, se comprueba si hay algún contenido que pueda ser reemplazado. Para que esto ocurra, la frecuencia de acceso del contenido candidato tiene que ser superior (más un cierto margen de control) a alguno de los contenidos de la cache cooperativa local.

Una vez se dispone de la suficiente capacidad disponible se envía al servidor remoto una

```
// Comprobar si el contenido es candidato a la cache autónoma?
If (ContenidoCacheAutonoma (videoId)==Cierto) then
    return(Falso);
// Suficiente ancho de banda para recibir el contenido?
if (SuficienteAnchoBandaEntrada(videoId)==Falso) then
    return(Falso);
// Comprobar si el contenido es candidato a la cache Cooperativa?
if (ContenidoCacheCooperativa(videoId)==Falso) then
    return(Falso);
// Suficiente almacenamiento para guardar el contenido?
if (GetTamañoContenido(videoId)<=GetCapacidadLibreCacheCooperativa())
    return(Cierto);
// Reemplazar contenido cache cooperativa?
return (ReemplazarContenidoCacheCooperativa(videoId));
```

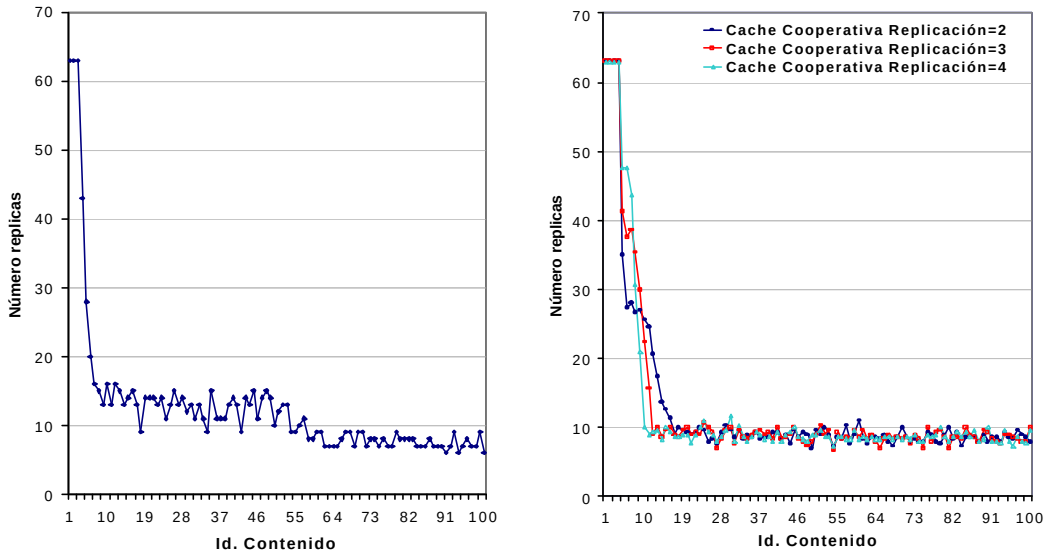
petición para que retransmita el contenido al servidor local (normalmente mediante un canal multicast compartido con el usuario local).

Efectividad de la cache cooperativa

Para mostrar la influencia de la cache cooperativa sobre el número de contenidos replicados en el sistema, la figura 6-11 muestra el número de copias existentes de cada uno de los contenidos del catálogo en un sistema Double P-Tree compuesto por 63 servidores-proxy.

La figura 6-11a muestra la replicación de los videos utilizando únicamente los esquemas de cache autónoma y mirroring, mientras que la figura 6-11b muestra las mismas estadísticas pero ahora utilizando también el esquema de cache cooperativa con capacidad suficiente para almacenar un video y modificando el número de replicas (*R*).

Con el esquema de caching puro los videos más populares aparecen replicados en todos los servidores locales (63 en total), el resto de videos reducen su replicación de forma brusca para estabilizarse alrededor de las 16 y 7 copias para cada contenido. Esta distribución de los contenidos implica que salvo los primeros 6 videos (de los cuales 3 pertenecen al primer nivel de cache) el resto aparece raramente replicados en el mirror distribuido lo cual no permite la activación de la política de balanceo de peticiones.



(a) Sin cache cooperativa. (b) Con cache cooperativa.

Figura 6-11. Replicación de contenidos en un sistema Double P-Tree

Al utilizar el esquema de caching cooperativo (figura 6-11b) la transición entre los videos más populares y el resto de videos del sistema no es tan brusca. El uso de esta política permite la aparición de una zona intermedia, en forma de escalón, entre los videos que forman los esquemas de caching (más populares) y mirorring (menos populares). La altura y anchura de este escalón depende de la capacidad y el número de replicados de la cache cooperativa. A medida que se incrementa el número de replicados el escalón es más alto y más estrecho (siempre que no se modifique la capacidad). Los videos intermedios son los que aumentan la replicación y con ello, la capacidad de activación de la política de balanceo de peticiones.

El incremento de la replicación se ve reflejado en la capacidad de activación de la política de balanceo de peticiones, tal y como muestra la tabla 6-5.

Tabla 6-5. Distribución de las peticiones en un sistema Double P-Tree con cache cooperativa

Peticiones					
Locales	**% Total**	**Remotas**	**% Total**	**Remotas susceptibles de ser balanceadas**	**% Total**
22.610	51,16%	14174	48,84%	7409	16,76%

Con la política de caching cooperativo el número de veces que se activa la política de balanceo se incrementa hasta las 7.409, el 17% de todas las peticiones del sistema (a diferencia de los datos mostrados en la tabla 6-4 cuando no se utilizaba la cache cooperativa: 6.122 peticiones, un 14% de todas las peticiones del sistema) ó que es lo mismo el 58% de todas las peticiones remotas.

En la tabla 6-6, mostramos la mejora obtenida en la política de balanceo de peticiones cuando se utiliza el esquema de cache cooperativa.

Tabla 6-6. Desbalanceo del tráfico utilizando caching cooperativo

Política de balanceo de tráfico	Distancia media de servicio	Desequilibrio del tráfico en los puertos	
		Global	Topología
Cache autónoma			
Balanceo de peticiones Gradual	1,742	17,79%	7,10%
Cache autónoma + Cache cooperativa			
Balanceo de peticiones Gradual (Rep=2)	1,743	15,40%	4.39%
Balanceo de peticiones Gradual (Rep=3)	1,743	16,26%	5,38%
Balanceo de peticiones Gradual (Rep=4)	1,743	17,31%	6,45%

Podemos comprobar que la utilización de la política de balanceo de peticiones en conjunción con el caching cooperativo logra minimizar el desbalanceo existente en las redes del sistema. Un desequilibrio medio del 14% en el trafico de la red (debido en su mayor parte a la diferencia de tráfico entre los puertos de usuario y el resto de puertos y un desequilibrio en los puertos de la topología apenas del 4% que nos permite afirmar que hemos alcanzado nuestro objetivo de lograr una distribución de tráfico balanceada en las redes de los sistemas P-Tree.

En la misma tabla también podemos comprobar los resultados obtenidos en función del número de replicados de la política de caching (parámetro R). La configuración que permite únicamente 2 replicados dentro de la cache cooperativa, es la que obtiene mejores resultados. Esto es consecuente con nuestras previsiones, ya que esta configuración es la que permite la replicación de un mayor número de contenidos diferentes y aumentar la capacidad de activación de la política de balanceo.

6.4 Evaluación de las políticas de balanceo

En este apartado evaluaremos el resultado de las distintas políticas de balanceo estáticas y dinámicas sobre el rendimiento de los sistemas Double P-Tree.

En la obtención de estos resultados hemos asumido una configuración base compuesta por un sistema Double P-Tree formada por 63 servidores locales con 3 puertos de servicio y una topología con tres hermanos. Cada red local dispone de un ancho de banda de 100 Mb/s por puerto, y los servidores locales un ancho de banda de servicio de 300 Mb/s (100Mb/s por cada uno de los puertos de servicio). Para la configuración de la políticas multicast se asume la utilización de una política de patching, y un buffer en los clientes con capacidad suficiente para 5 minutos de video.

6.4.1 Evaluación de las políticas de balanceo estáticas

La figura 6-12 muestra el ancho de banda efectivo obtenido por la arquitectura Double P-Tree para las distintas políticas de balanceo estáticas.

Para comprobar la influencia de la capacidad de almacenamiento sobre la efectividad de dichas políticas se muestran los resultados variando la capacidad de almacenamiento disponible en los servidores locales (rango 5% a 40% de los contenidos del catálogo).

Los resultados muestran que las políticas de asignación de videos equilibrados obtienen siempre el mejor rendimiento, lo cual pone de manifiesto la influencia del desbalanceo del tráfico

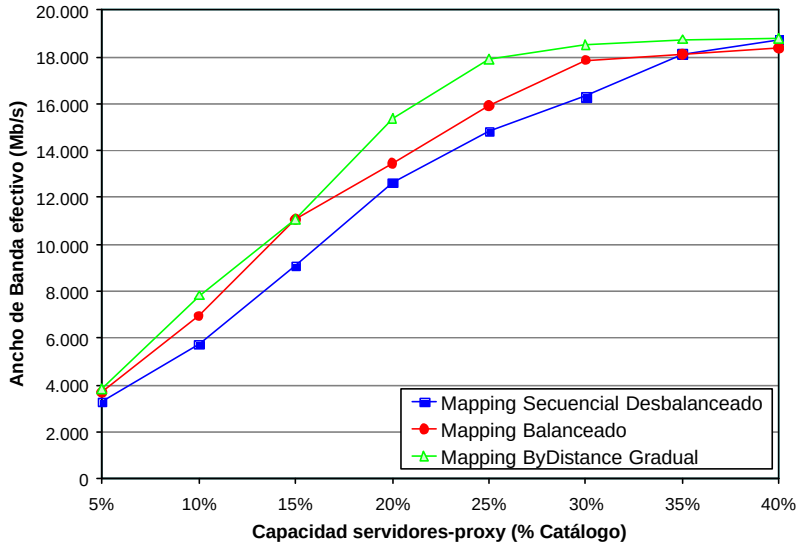


Figura 6-12. Rendimiento de las políticas estáticas de balanceo del tráfico

sobre la eficiencia de las arquitecturas distribuidas de VoD.

Se puede observar también, que el rendimiento de las políticas estáticas de balanceo de tráfico depende de la capacidad de almacenamiento de los servidores-proxy. Cuando los servidores tienen un capacidad pequeña, las políticas estáticas tienen un mayor impacto sobre la distancia media de servicio y, por lo tanto, el incremento de rendimiento asociado con estas políticas es pequeño. Estas políticas muestran su mayor efectividad en los rangos intermedios de almacenamiento (capacidad servidores-proxy entre 10% y 25%), debido a que con estas configuraciones el tráfico recibido por los puertos de la topología son los que finalmente definen los requisitos de ancho de banda de las redes locales.

Sin embargo, cuando el servidor local dispone de suficiente capacidad como para almacenar el 40% de los videos del sistema, la eficiencia de las políticas de balanceo no se ven reflejados en una mejora de los resultados del sistema. Este efecto se debe a que el incremento de la capacidad de los servidores incrementa la probabilidad de que las peticiones sean servidas localmente, reduciendo el tráfico que circula por los puertos de la topologías. Esta modificación en la distribución del tráfico en las redes locales provoca que los requisitos de las mismas ya no dependan de los puertos de la topología si no de los puertos de servicio. Por lo tanto, un tráfico más balanceo en los primeros no consigue reducir los requisitos de ancho de banda de las redes del sistema.

De las dos políticas estáticas de balanceo de tráfico, los mejores resultados se obtienen mediante la política de asignación de videos ByDistance Gradual. La utilización de esta política permite incrementar la capacidad de servicio de la arquitectura Double P-Tree en alrededor del 11%. Sin embargo, la política de mapping Balanceado, aunque logra un tráfico más equilibrado, éste no es suficiente para compensar la perdida de rendimiento asociada con la deterioro de la distancia media de servicio.

Es importante hacer notar que la mejora del rendimiento obtenida mediante la política ByDistance Gradual no se puede atribuir a la distancia de servicio (ya que como vimos en la tabla 6-2 la distancia de servicio obtenida es prácticamente idéntica), sino a una mejora de los requisitos de ancho de banda de red obtenidos gracias al balanceo del tráfico.

6.4.2 Evaluación de la política de balanceo de peticiones

La figura 6-13 muestra el rendimiento de la política de balanceo de peticiones. Como referencia, en la figura también se muestra el rendimiento de la mejor política estática de balanceo de tráfico (ByDistance Gradual).

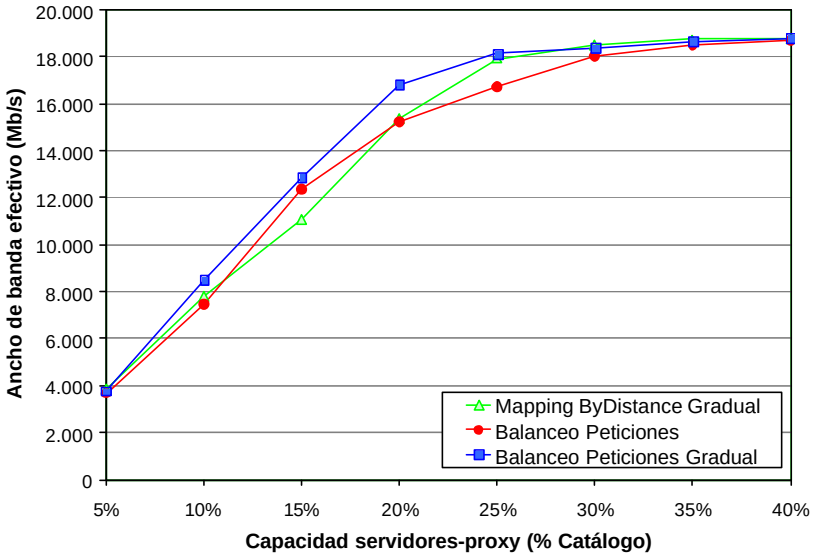


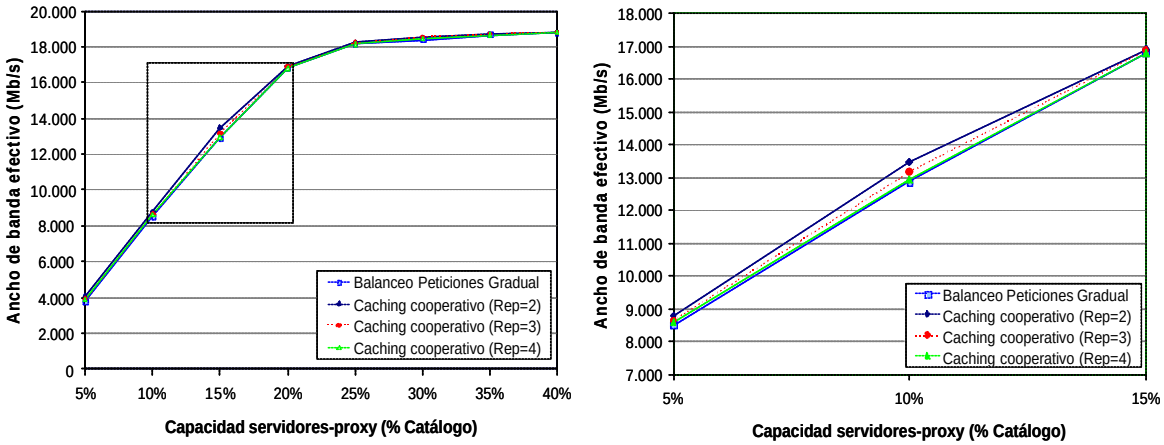
Figura 6-13. Rendimiento de las políticas de balanceo de peticiones

Podemos comprobar que la política de balanceo de peticiones por si sola, logra incrementar la capacidad de servicio de la arquitectura Double P-Tree en alrededor de un 11% (respecto al mismo sistema sin utilizar una política de desbalanceado) y obtiene resultados similares ó mejores a las políticas estáticas de balanceo de tráfico.

Estos resultados también nos permiten comprobar la viabilidad de la colaboración de las políticas estáticas con las políticas dinámicas, con el objetivo de reducir el desequilibrio de tráfico de las redes. La utilización conjunta de las políticas de balanceo de peticiones y ByDistance Gradual es la que obtiene los mejores resultados. Esta configuración logra mejorar el rendimiento en alrededor de un 10% con respecto al obtenido por cada una de las políticas utilizadas de forma individual.

6.4.3 Efecto de la cache cooperativa sobre la política de balanceo de peticiones

En este apartado evaluamos la influencia de la utilización del esquema de caching cooperativo sobre el rendimiento de la política de balanceo de peticiones. Como vimos en el apartado 6.3, el principal objetivo de la cache cooperativa es incrementar el margen de maniobra de la política de balanceo de peticiones mediante el incremento del número de contenidos replicados en el mirror distribuido.



(a) Escala normal **(b)** Ampliación

Figura 6-14. Rendimiento de la política de balanceo de peticiones utilizando caching cooperativo

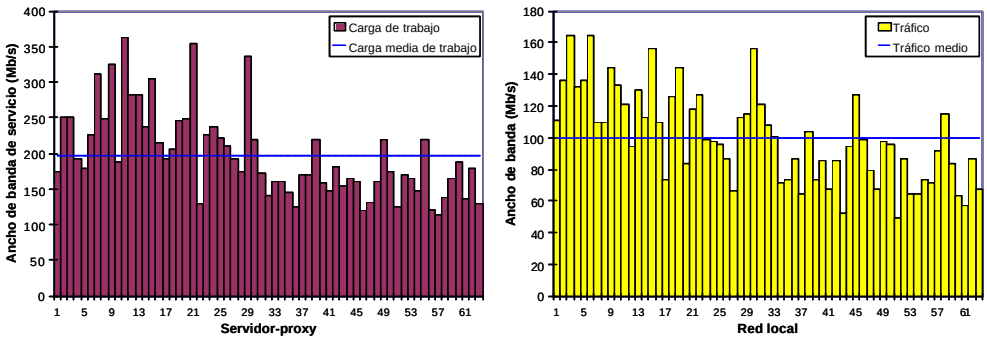
La figura 6-14 muestra el rendimiento de la política de balanceo de peticiones cuando se utiliza la cache cooperativa y un factor de replicación de 2, 3 y 4 contenidos replicados. Estos resultados se han obtenido utilizado una capacidad para la cache de cooperativa suficiente para almacenar un contenido multimedia. Una capacidad mayor no resulta efectiva debido a que aunque se mejora el balanceo del tráfico, la excesiva replicación introducida afecta a la distancia media de servicio y con ello a la capacidad de servicio del sistema.

En la figura 6-14b se puede comprobar que en el rango 10%-20%, la configuración que obtiene los mejores resultados es la que utiliza únicamente un factor de replicación de dos contenidos dentro de la cache cooperativa. Esta configuración logra un incremento adicional del rendimiento del alrededor del 5% respecto a la configuración que no utiliza la cache cooperativa para mejorar la efectividad de la política de balanceo de peticiones.

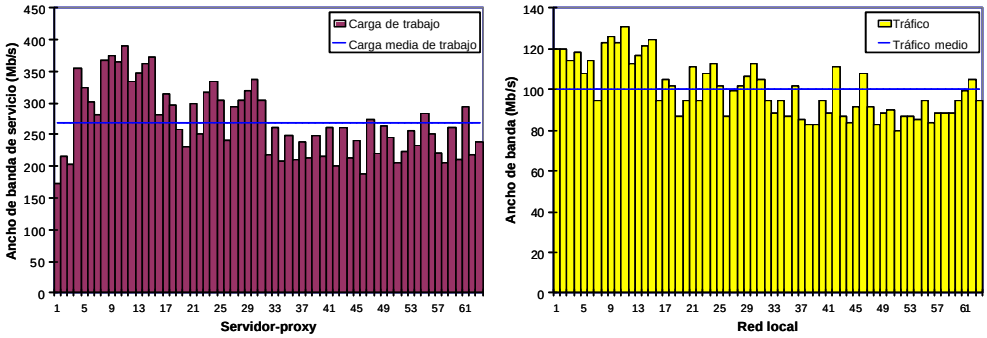
6.4.4 Influencia de las políticas de balanceo del tráfico sobre la distribución de carga en el sistema

Para finalizar con la evaluación de las políticas de balanceo del tráfico en las redes, vamos a estudiar el efecto de estas políticas sobre el desbalanceo global de la carga recibida por los servidores-proxy y las redes locales, y por tanto, sobre el rendimiento.

La figura 6-15 muestra la distribución de la carga y el tráfico entre los 63 servidores-proxy y redes locales de una arquitectura Double P-Tree.



(a) Sin utilizar políticas de balanceo de tráfico



(b) Utilizando políticas de balanceo de tráfico

Figura 6-15. Rendimiento de las políticas estáticas de balanceo del tráfico

La figura 6-15a muestra la distribución de la carga en un sistema en el cual no se utilizan políticas de balanceo del tráfico en las redes locales.

La gráfica de la derecha muestra la distribución de la carga del sistema entre los servidores. Se puede observar como la carga de los distintos servidores está muy desequilibrada, mientras la carga media del sistema por servidor es de 200 Mb/s, la carga máxima (366 Mb/s) se localiza en el servidor número 11 y la carga mínima es de 114 Mb/s localizada en el servidor número 56. La desviación estándar de la carga en los servidores del sistema es de 61Mb/s, lo cual implica un desbalanceo del 30% sobre la carga media del sistema.

En la gráfica de la izquierda mostramos la distribución del tráfico en las redes locales de la arquitectura. Al igual que ocurre con la carga de los servidores, el tráfico de las redes locales también está fuertemente desbalanceado. El tráfico medio de las redes locales es de 100 Mb/s, el tráfico máximo es de 165 Mb/s y el tráfico mínimo de 50 Mb/s. La desviación estándar del tráfico en las redes alcanza los 29 Mb/s, lo cual implica un desbalanceo del 29% sobre el tráfico medio del sistema, el cual es bastante similar al desbalanceo de los servidores-proxy.

La figura 6-15b muestra la distribución de la carga en un sistema en el cual se han utilizado las políticas ByDistante Gradual y de balanceo de peticiones para equilibrar el tráfico en las redes locales. A primera vista se puede comprobar como el desbalanceo global en los servidores-proxy y las redes locales es considerablemente menor en este segundo caso.

Al utilizar las políticas de balanceo del tráfico de las redes, la carga media del sistema aumenta hasta los 270 Mb/s, la carga máxima es de 390 Mb/s y la carga mínima es de 174 Mb/s. La desviación estándar de la carga en los servidores del sistema es de 54Mb/s, lo cual implica un desbalanceo de únicamente el 20% sobre la carga media del sistema, frente al 30% obtenido cuando no se utilza las políticas de balanceo.

Con respecto al desbalanceo global del tráfico de las redes locales se obtienen resultados similares. El tráfico medio de las redes locales es de 100 Mb/s, el tráfico máximo es de 130 Mb/s y el tráfico mínima de 80 Mb/s. La desviación estándar del tráfico en las redes alcanza los 13 Mb/s, lo cual implica un desbalanceo del 13% sobre el tráfico medio del sistema (frente al 29% cuando no se utilza las políticas de balanceo).

6.5 Conclusiones

En este capítulo hemos analizado la influencia sobre el rendimiento del sistema LVoD de la distribución de la carga / tráfico entre los distintos componentes de la arquitectura.

El desbalanceo global entre la carga y el tráfico recibido se puede atribuir principalmente a la diferente conectividad de los servidores del sistema, a la distribución no uniforme de las peticiones de los usuarios ó a la distribución de los contenidos entre los distintos mirrors de los servidores-proxy.

A nivel de las redes, otra posible causa de desbalanceo procede de la distribución del tráfico de cada red local entre los distintos puertos de la misma. Debido a las características de las redes segmentadas, el desbalanceo entre el tráfico de los puertos puede provocar una reducción en la eficiencia de las arquitecturas distribuidas, incrementado los requisitos de ancho de banda de red y disminuyendo la capacidad de servicio del sistema.

El desbalanceo en los puertos de las redes se puede atribuir al desequilibrio existente entre los distintos tipos de puertos: puertos de servicio, puertos de la topología y puertos de los usuarios. El tráfico asociado con los puertos de servicio y puertos de usuarios no puede ser redistribuido sin modificar la arquitectura. Por lo tanto, para poder reducir el desbalanceo de las redes locales y con ellos los requisitos de ancho de banda de red del sistema, la solución más factible consiste en equilibrar el tráfico asociado con los puertos de la topología. La distribución

del tráfico entre los puertos de la topología depende en última instancia de la distribución de las peticiones que no se pueden servir localmente entre los distintos servidores remotos. La distribución de estas peticiones depende a su vez de cómo se haya distribuido los contenidos entre los distintos servidores-proxy.

Se han propuesto dos tipos de políticas diferentes para evitar el desbalanceo en las redes locales: políticas de balanceo estáticas y políticas de balanceo dinámicas.

Las políticas estáticas (mapping balanceado y ByDistance gradual) modifican la política de asignación ByDistance de forma que se logre que los contenidos asignados a cada uno de los servidores-proxy estén más equilibrados entre si. Estas políticas han demostrado su efectividad en la reducción del desbalanceo de las redes, sin embargo, debido a su naturaleza estática (que reduce su capacidad de maniobra) no consiguen eliminar totalmente el desbalanceo en los puertos de la topología.

El método de balanceo de peticiones es una política dinámica que permite redistribuir las peticiones remotas una vez éstas se generan, de forma que se logre un tráfico más balanceado entre los distintos puertos de la topología. Para incrementar la eficiencia de la política de balanceo de peticiones, es clave aumentar la replicación (de forma controlada) de los contenidos en el sistema LVoD mediante la utilización de una cache cooperativa.

Ambas políticas (dinámicas y estáticas) han demostrado ser compatibles y beneficiosas, pues su utilización conjunta permite la reducción del desbalanceo del tráfico en los puertos de la topología hasta únicamente el 4%. Este desbalanceo se puede considerar residual. Mediante la consecución de un tráfico balanceado en las redes locales permite incrementar la capacidad de servicio de las arquitecturas Double P-Tree en más de un 25%.

Como líneas de futura investigación dentro de este ámbito proponemos el equilibrio del tráfico de los restantes puertos de las redes (servicio y usuarios) mediante políticas que permitan desviar tráfico de los puertos de servicio (normalmente los más saturados) a los puertos de usuario. Este objetivo se puede lograr incrementado la distribución de la arquitectura, permitiendo que los mismos usuarios puedan servir peticiones a otros usuarios mediante la utilización de políticas de *chaining*.

Otra futura línea de investigación está orientada a analizar y tratar de solventar el desbalanceo global que se puede producir en las arquitecturas distribuidas, mediante la modificación de la política de balanceo de peticiones. Para afrontar este problema, la política debería tener en cuenta el desequilibrio de la carga entre los servidores-proxy a la hora de tomar la decisión de quién tiene que servir una petición remota.

Capítulo 7

Análisis del rendimiento de las arquitecturas LVoD distribuidas

Resumen

En este capítulo analizamos la influencia de las políticas de multicast sobre las distintas arquitecturas propuestas. A continuación, a modo de recapitulación, realizaremos un estudio comparativo entre las arquitecturas propuestas (Proxy-Tree y Double P-Tree) y las arquitecturas alternativas de LVoD. En este análisis, se tendrán en cuenta las nuevas políticas propuestas, así como la utilización de políticas de servicio unicast y multicast. Por último, se analiza la influencia en el rendimiento de las arquitecturas P-Tree de cada una de las propuestas realizadas durante la tesis.

7.1 Introducción

En este capítulo procederemos a la evaluación de la eficiencia de las arquitecturas P-Tree teniendo en cuenta las optimizaciones introducidas en los capítulos anteriores y la utilización de políticas de servicio multicast.

Esta evaluación se realiza aprovechando el amplio espectro de pruebas que podemos realizar sobre las distintas arquitecturas gracias al entorno de simulación presentado en el cuarto capítulo.

Los objetivos de este capítulo son los siguientes:

1. Analizar la influencia de las políticas de servicio multicast sobre el rendimiento de las arquitecturas propuestas.
2. Comparar el rendimiento / eficiencia de las distintas arquitecturas LVoD.
3. Mostrar la influencia de los distintas contribuciones realizadas sobre la eficiencia de la arquitecturas P-Tree.

7.2 Eficiencia de las políticas de servicio multicast en la arquitectura Double P-Tree

Tal y como hemos comentado a lo largo de la tesis, las técnicas de compartición de recursos entre peticiones son uno de los principales pilares sobre los que se sustenta la efectividad final de los sistemas VoD.

Dentro del rango de técnicas de compartición de recursos, las políticas de servicio multicast ocupan un lugar preponderante. Estas técnicas tienen como principal ventaja el permitir compartir los recursos requeridos por una petición a lo largo de todo el camino de servicio (sistema de almacenamiento, buffers intermedios en memoria, ancho de banda de red, etc...).

De todas las políticas de multicast propuestas en la literatura nosotros nos hemos centrado en la utilización de la política de patching, debido a las siguientes consideraciones:

- Permite ofrecer servicios de video-bajo-demanda verdadero.
- Es una de las políticas multicast más sencilla de implementar.

- Exige pocos requisitos al STB del usuario, básicamente que éste sea capaz de recibir simultáneamente dos streams de video y un buffer para almacenar una pequeña parte del video a reproducir.
- Permite obtener una gran eficiencia con muy pocos recursos.

En los siguientes apartados se expone con más detalle el funcionamiento de esta política y los principales parámetros que afectan a su rendimiento.

7.2.1 Patching

Tal y como comentamos en el capítulo 1, la política de patching [Hua98] [Sen99] [Cai99] es una política de servicio multicast que soporta servicios de VoD verdadero. Esta política se basa en aprovechar los flujos de información que se están transmitiendo a otros usuarios para reducir el consumo de ancho de banda de red y de servicio de las nuevas peticiones.

Cuando el servidor recibe una nueva petición se comprueba si existe algún stream que esté transmitiendo el mismo contenido y que esté dentro de la ventana de tiempo del buffer del nuevo usuario. Si esta condición se cumple, el nuevo usuario se añade al canal de transmisión activo, asegurándose la recepción del resto del video de forma compartida con la primera petición utilizando un canal multicast.

No obstante, para que el usuario pueda reproducir correctamente el video, hay que proporcionarle el fragmento inicial. Esta porción del video (patch) se debe transmitir mediante un canal independiente, denominado canal de patch. Una vez llega este canal al usuario, este ya puede empezar inmediatamente la reproducción del video.

En la figura 7-1 mostramos mediante un ejemplo el funcionamiento de la política de patching. En el ejemplo se supone que los buffers de usuario tienen capacidad para 15 minutos de video. Se supone que transcurridos 12 minutos ha llegado una petición del usuario 1 para el video j, la cual ha sido servida mediante un canal unicast.

Al cabo de 5 minutos se recibe una petición para el mismo video por parte del usuario 2. Esta petición se puede servir mediante patching debido a que el buffer de usuario tiene suficiente espacio como para almacenar al mismo tiempo la porción del video que falta (minutos 0-5) y la información del canal multicast (desde el minuto 5). Por lo tanto, se añade el usuario 2 al canal de video recibido por el primer usuario (el cual ahora se convierte en un canal multicast) y se crea un nuevo canal unicast para enviar al usuario 2 los primeros 5 minutos del video.

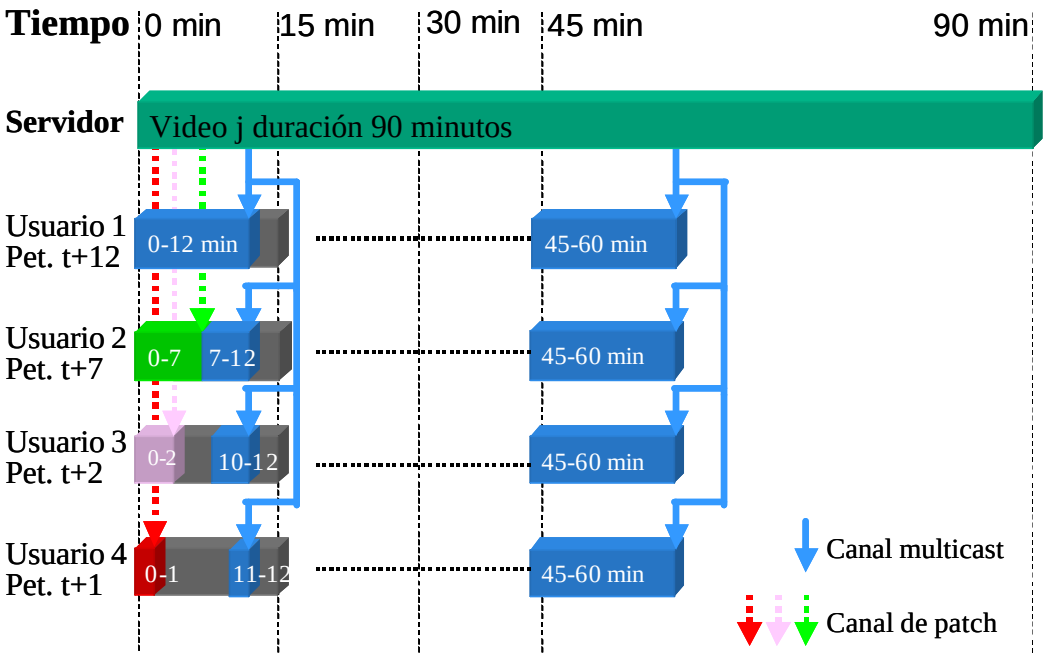


Figura 7-1. Funcionamiento de la política de patching

El mismo procedimiento se sigue con las peticiones de los usuarios 3 y 4 que llegan al cabo de 5 y 6 minutos. Estos dos usuarios recibirán mediante sendos canales de patch los primeros 10 y 11 minutos del video y el resto del video lo recibirán del canal multicast.

Mediante la política de *patching* los usuarios 2, 3 y 4 han podido compartir la mayor parte de los recursos utilizados por el usuario 1, únicamente requiriendo canales independientes para recibir los primeros 7, 10 y 11 minutos del video respectivamente. El ahorro de recursos de estos usuarios alcanza el 90% del total.

7.2.2 Influencia del tamaño del buffer de los STB de los clientes

La influencia del tamaño del buffer de los clientes sobre la eficiencia de la política de patching la vamos a estudiar, analizando como evoluciona el número de peticiones que puede soportar el sistema LVoD a medida que se incrementa el tamaño del buffer.

El tamaño del buffer de los clientes lo vamos a especificar en función del número de minutos de un video que es capaz de almacenar. En concreto vamos a probar tamaños del buffer capaces de almacenar desde 1 a 15 minutos del video transmitido.

Este análisis se ha realizado utilizando una arquitectura Double P-Tree con 7 hermanos y 63 redes locales. La figura 7-2 muestra el número de peticiones gestionadas por el sistema LVoD

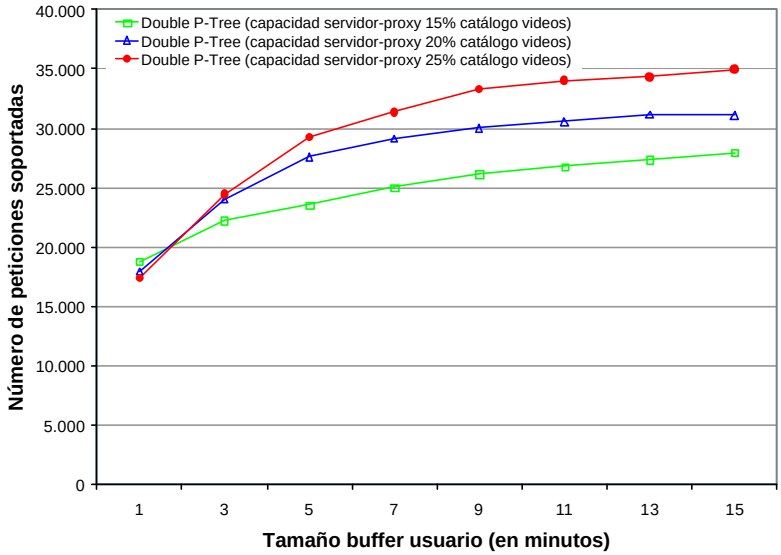


Figura 7-2. Número peticiones soportadas & el tamaño del buffer de usuario

(sin que se produzca denegación de servicio) y utilizando distintas capacidades de almacenamiento en los servidores-proxy (15%, 20% y 25% de los contenidos del catálogo).

En la figura se puede comprobar que a medida que se incrementa la capacidad de almacenamiento disponible en los STB de los usuarios también se aumenta la efectividad de la política multicast y, por lo tanto, la capacidad de gestión de peticiones por parte del sistema LVoD.

A partir de un tamaño de buffer con capacidad para 13 minutos de video la efectividad obtenida gracias al nuevo almacenamiento añadido ya no es tan grande. Este efecto es debido a que el ahorro de recursos asociado con estas peticiones también es menor y que la mayoría de peticiones dirigidas a los videos más populares ya son capturadas con buffers más pequeños.

7.2.3 Influencia del volumen de peticiones recibidas

La frecuencia de llegada de nuevas peticiones por parte de los usuarios también tiene un impacto significativo sobre el rendimiento de las políticas multicast en general y sobre la política de patching en concreto.

Esta influencia se explica en base a que para poder servir conjuntamente 2 peticiones, éstas tienen que llegar al sistema lo suficientemente cercanas en el tiempo (dependiendo del tamaño del buffer) para que el segundo usuario pueda dar cabida en su buffer la primera porción del

video transmitida por el canal multicast y todo el fragmento inicial del video que se tiene que recibir mediante el canal de patch.

Al incrementar el número de peticiones que se producen en el sistema, se aumenta la probabilidad de que dos peticiones a un mismo contenido se puedan servir conjuntamente utilizando la políticas de patching.

La figura 7-3 muestra el número de peticiones gestionadas por la arquitectura Double P-Tree en función de la frecuencia de llegada de las peticiones a cada una de las redes locales del sistema LVoD. Como era de esperar, los resultados muestran un incremento del número de peticiones gestionadas por el sistema a medida que se aumenta la carga recibida (número de peticiones recibidas por minuto).

7.3 Análisis comparativo de las arquitecturas LVoD

En este apartado vamos a proceder a comparar las arquitecturas P-Tree con las arquitecturas distribuidas propuestas en la literatura para implementar un sistema LVoD.

Durante este estudio vamos a utilizar el ancho de banda efectivo como principal métrica de comparación. Este análisis, nos permitirá evaluar mediante la simulación, la capacidad máxima de servicio que puede soportar las diferentes arquitecturas distribuidas: servidores-independientes, un nivel de servidores-proxy, Proxy-Tree y Double P-Tree. Al evaluar las arquitecturas P-Tree utilizaremos las distintas políticas de optimización para la gestión del almacenamiento y el balanceo del tráfico presentadas en los dos capítulos anteriores.

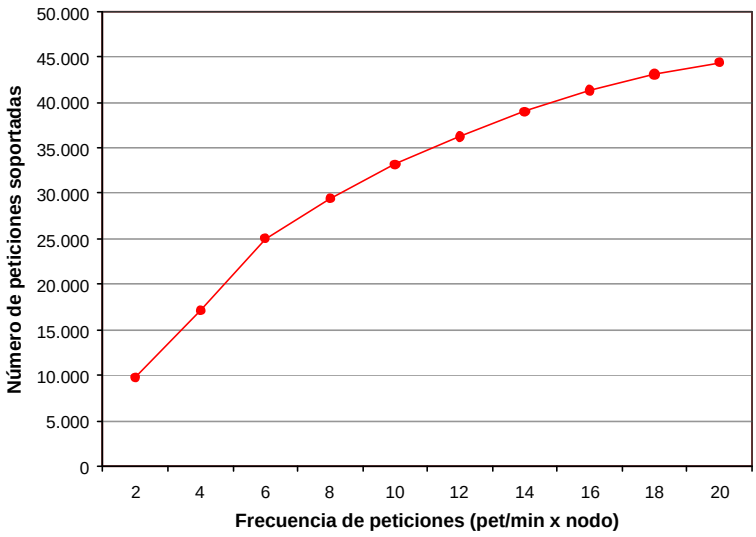


Figura 7-3. Número peticiones soportadas & la frecuencia de llegada de peticiones

Para obtener estos resultados, se han utilizado los parámetros mostrados en la tabla 7-1, asignado un ancho de banda de red total de 6.300 Mb/s a distribuir entre las todas las 63 redes del sistema y un ancho de banda de servicio total de 6.300 Mb/s (cuando se utiliza un puerto de servicio) ó 18.900 Mb/s (con 3 puertos de servicio) que se distribuye entre todos los servidores-proxy. Para obtener la capacidad máxima de servicio, hemos saturado el sistema utilizando una alta frecuencia de llegada de peticiones y simulamos el comportamiento del sistema hasta que el ancho de banda de red disponible se agota. En este instante, el sistema alcanza su máxima capacidad y procedemos a evaluar sus principales métricas de rendimiento. Este análisis se ha realizado, utilizando indistintamente políticas servicio unicast, como la técnica patching.

Tabla 7-1. Principales parámetros de simulación

Parámetro	Valor	Parámetro	Valor
· Numero de videos	100	· Técnica multicast	Patching
· Longitud video	90 minutos	· Tamaño buffer usuario	5 minutos
· Redes locales	63	· Frecuencia llegada peticiones (λ)	10 pet/min x nodo
· Ancho de banda red	100 Mb/s		
· Ancho de banda servicio 1 puerto de servicio (1P)	100 Mb/s	· Distribución de Poisson	$p_i = \frac{l_i^k}{k!} e^{-l_i}$ (1)
· Ancho de banda servicio 3 puerto de servicio (3P)	300 Mb/s	· Distribución de Zipf (factor skew, z = 0,728)	$p_x = \frac{1}{x^z \cdot \sum_{i=1}^{Sv} 1/i^z}$ (2)
· Capacidad almacenamiento	25 videos		

7.3.1 Rendimiento de las arquitecturas distribuidas con políticas unicast

Utilizando políticas de servicio unicast, en la figura 7-4, podemos ver que la arquitectura de servidores independientes es la que obtiene los mejores resultados, alcanzado la capacidad máxima de servicio teórica de 18.800 Mb/s. Mientras tanto la arquitectura Double P-Tree con 7 hermanos obtiene una ancho de banda efectivo de 18.400 Mb/s y una capacidad de servicio de 18.000 Mb/s cuando se utilizan una topología con tres hermanos (un 2% y un 4% menos respectivamente).

Estos resultados son debidos al ancho de banda adicional requerido para servir las peticiones remotas y a los menores requisitos de almacenamiento de la arquitectura Double P-Tree (cuatro veces menos). No obstante, esta reducción en el rendimiento es considerablemente menor a lo que correspondería si tuviésemos en cuenta la distancia media de servicio de ambas arquitecturas (1 para la arquitectura de servidores independientes frente a una distancia de

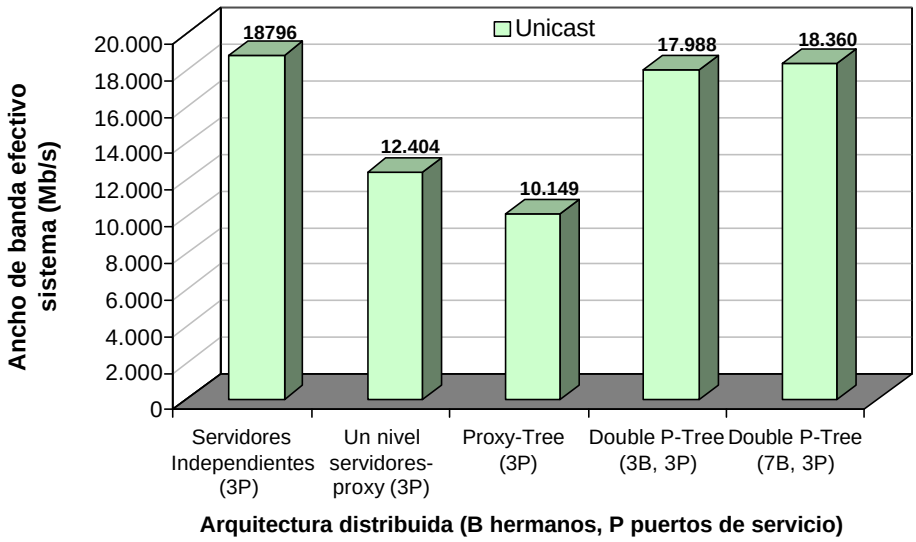


Figura 7-4. Rendimiento de las arquitecturas LVoD distribuidas utilizando técnicas unicast

servicio de 1.58 y 1.48 para las dos configuraciones Double P-Tree). Estos resultados demuestran la efectividad de nuestra arquitectura en distribuir y balancear el tráfico entre los puertos de la topología para reducir los requisitos de red.

También podemos comprobar que la arquitectura Double P-Tree supera la arquitectura de servidores-proxy de un nivel en más de un 45%. Respecto a la arquitectura P-Tree, podemos observar la combinación de una menor conectividad, una capacidad de almacenamiento reducida y una saturación de las redes debido al alto ancho de banda de servicio, provoca que esta arquitectura sea la que menor rendimiento obtiene. La razón de este comportamiento estriba en la saturación de los puertos de la topología, que aumenta considerablemente los requisitos de ancho de banda de las redes locales de la arquitectura P-Tree.

7.3.2 Rendimiento de las arquitecturas distribuidas con políticas multicast

El sistema Double P-Tree demuestra todo su potencial como arquitectura LVoD cuando se utilizan políticas de compartición de recursos.

La figura 7-5 muestra los resultados obtenidos por las distintas arquitecturas distribuidas cuando se utiliza la política de servicio Patching. En este escenario más realista la arquitectura Double P-Tree utilizando 7 hermanos se muestra como la mejor solución, mejorando en más de un 47% la capacidad de servicio de la arquitectura de servidores independientes (44.3 Gb/s frente a 30 Gb/s), en un 190% el rendimiento de la arquitectura de servidores-proxy de un nivel

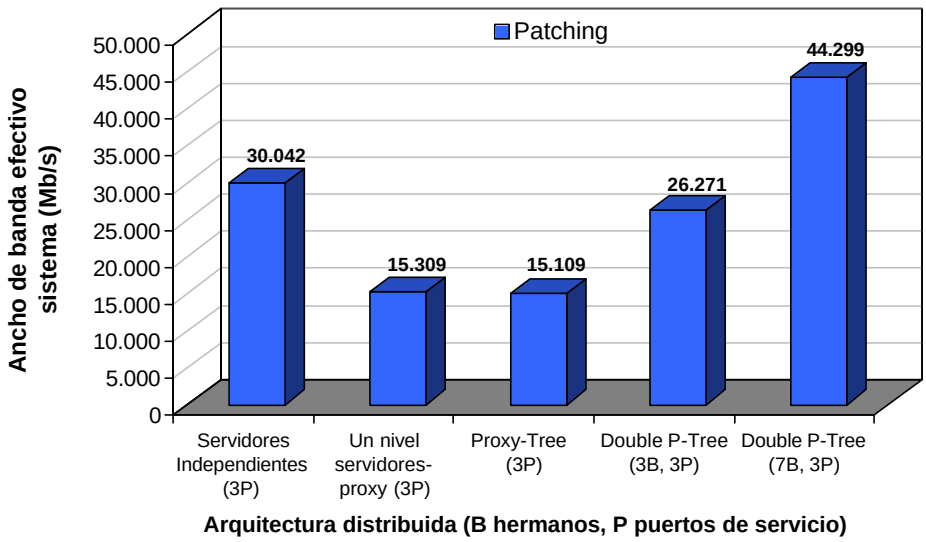


Figura 7-5. Rendimiento de las arquitecturas LVoD distribuidas utilizando técnicas multicast

(44.3Gb/s rente a 15.3Gb/s) y en un 194% el rendimiento de la arquitectura P-Tree (44.3Gb/s con respecto a 15.1Gb/s).

El principal argumento para este espectacular rendimiento de la arquitectura Double P-Tree estriba en su mejor conectividad respecto a las otras arquitecturas distribuidas. Esta mejor conectividad significa que en la arquitectura Double P-Tree, un stream asociado con un video del mirror puede ser potencialmente compartido por todas las peticiones remotas provenientes de las redes adyacentes. Esto significa multiplicar la probabilidad de compartición por la conectividad de la topología. En el caso de la arquitectura Double P-Tree con siete hermanos, la probabilidad de compartición se multiplica por más de 9. Mientras tanto, en las arquitecturas poco interconectadas (servidores independientes y un nivel de servidores-proxy) el potencial de compartición de recursos queda limitado únicamente a las peticiones locales.

7.3.3 Influencia de la frecuencia de llegada de peticiones

En los resultados mostrados en la figura 7-5 ya hemos podido comprobar que la arquitectura Double P-Tree es la que obtiene un mejor rendimiento cuando se utilizan políticas multicast. Ya hemos visto también (en el apartado 7.2.2) que el rendimiento de las políticas multicast depende considerablemente de la frecuencia de llegadas de peticiones nuevas al sistema.

En esta sección vamos a analizar la relación existente entre la frecuencia de llegada de peticiones (volumen de carga del sistema LVoD) y las tres principales arquitecturas LVoD distribuidas: Proxy-Tree, Double P-Tree y servidores independientes.

Para este análisis, la metodología de experimentación ha variado ligeramente respecto al resto de secciones. En este caso, hemos querido comprobar como se comporta cada una de las arquitecturas a partir de unos recursos asignados estáticamente, durante un periodo de tiempo fijando de antemano (90 minutos). Bajo estas condiciones y dependiendo del volumen de carga, el sistema no podrá atender todas las peticiones que recibe y tendrá que denegar el servicio a las que no se pueda servir con los recursos disponibles.

La figura 7-6 muestra el número de peticiones que puede soportar el sistema LVoD para los tres sistemas analizados y a medida que variamos el número de peticiones recibidas por minuto en cada una de las redes. Se puede observar, tal y como se esperaba, que la arquitectura Double P-Tree es la que muestra una mayor capacidad de servicio a medida que se incrementa la carga recibida por el sistema.

La relevancia de estos resultados estriba en que nos permiten evaluar a partir de que volumen de carga la arquitectura Double P-Tree empieza a obtener mejores resultados que la arquitectura de servidores independientes. Tal y como se puede observar en la figura, este umbral se sitúa en las 4 peticiones por minuto. Cuando el volumen de carga no supera esta cota, la política de patching con un buffer de 5 minutos, difícilmente puede servir conjuntamente un número elevado de peticiones y, por lo tanto, la arquitectura Double P-Tree no puede aprovecharse de su mayor conectividad. Sin embargo, una vez superado este umbral, la arquitectura Double P-Tree supera

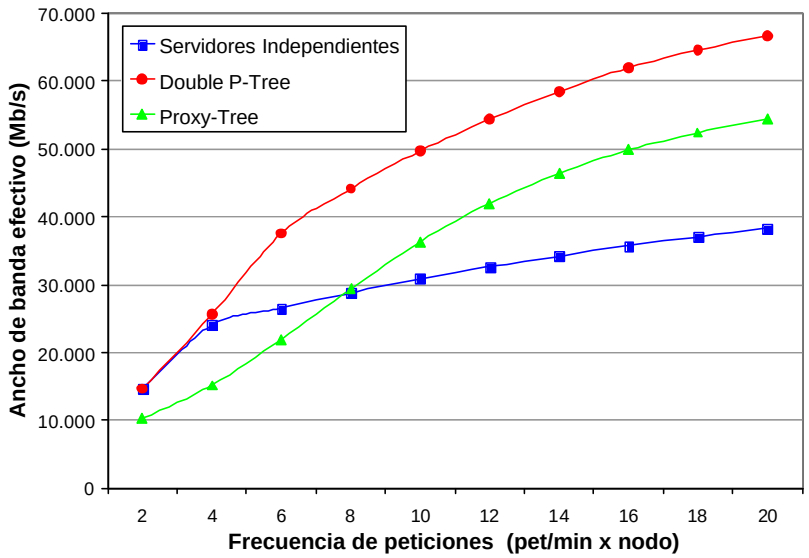


Figura 7-6. Rendimiento de las arquitecturas LVoD distribuidas & frecuencia peticiones

siempre la capacidad de servicio del resto de arquitecturas distribuidas.

Los resultados de la figura también muestran que la arquitectura P-Tree supera el rendimiento de la arquitectura de servidores independientes cuando la frecuencia de llegada de las peticiones supera las 8 por minuto. Estos resultados demuestran la mayor capacidad de las arquitecturas P-Tree para balancear la carga del sistema y distribuir el tráfico de las redes entre los distintos enlaces de la topología.

Finalmente, la figura 7-7 muestra como evoluciona el volumen de peticiones denegadas a medida que se aumenta el volumen de carga soportado por las tres arquitecturas. Los resultados mostrados son consecuentes con los mostrados en la figura anterior. La arquitectura Double P-Tree es la que rechaza menos peticiones, alrededor de un 18% menos que la arquitectura de servidores independientes y un 8% menos con respecto a la arquitectura P-Tree.

7.3.4 Influencia del número de puertos de servicio

La capacidad de servicio de los servidores de las redes locales es otro de los parámetros que se deben tener en cuenta al diseñar una arquitectura LVoD. En la medida que aumentamos la capacidad de servicio de los servidores, también incrementamos la carga de tráfico distribuida entre los distintos puertos de la red, así como la efectividad y complejidad del sistema.

El aumento del tráfico de la red es especialmente crítico para los puertos de la topología en las arquitecturas distribuidas. La combinación de un servidor con una alta capacidad de servicio y una capacidad de almacenamiento reducida puede aumentar considerablemente el tráfico remoto

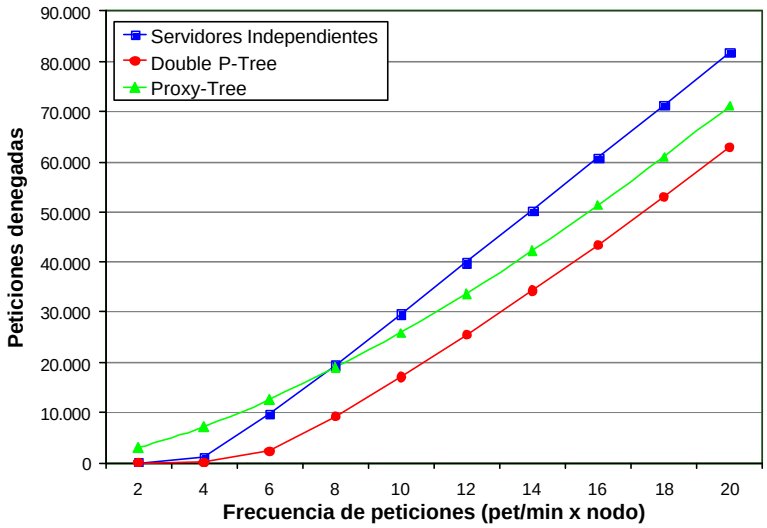


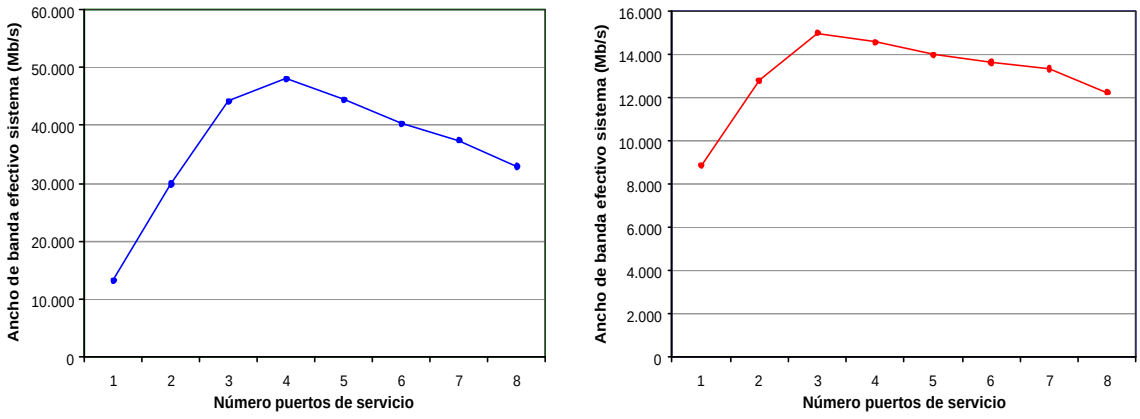
Figura 7-7. Denegación de servicio en las arquitecturas LVoD distribuidas & frecuencia peticiones

y puede fácilmente provocar la saturación de los enlaces de interconexión de la topología.

En la figura 7-8 analizamos la influencia de la capacidad de servicio de los servidores-proxy sobre el rendimiento de las arquitecturas P-Tree. El aumento de la capacidad de servicio lo vamos a realizar al mismo tiempo que incrementamos el número de puertos de servicio de los servidores-proxy. Se dispone de un ancho de banda de servicio de 100 Mb/s por cada puerto de servicio.

A medida que se aumenta la capacidad de servicio en la arquitectura Double P-Tree, figura 7-8a, podemos observar un aumento de la capacidad de servicio. Sin embargo, también podemos observar que el aumento de la capacidad de servicio no es proporcional. Mientras que con un puerto de servicio se obtiene un rendimiento de 13.2 Gb/s; al utilizar dos puertos de servicio se obtiene una capacidad de servicio del sistema de 29.9 Gb/s, más del doble (126%) que con un único puerto de servicio. Este aumento en el rendimiento del sistema es considerablemente mayor al que se debería haber obtenido debido al aumento de la capacidad de servicio de los servidores-proxy (63 servidores * 100 Mb/s = 6.3 Gb/s). Estas mayores prestaciones sólo se pueden justificar mediante la efectividad de las políticas de servicio multicast.

La progresión del rendimiento anterior se mantiene cuando se utilizan tres puertos de servicio (la capacidad aumenta en 14.2 Gb/s, hasta los 44.2 Gb/s). Sin embargo, al utilizar un mayor número de puertos de servicio empezamos a observar una progresiva desaceleración del aumento del rendimiento del sistema. Con 4 puertos de servicio la capacidad del sistema se aumenta en un 9% (hasta 48.1 Gb/s) y a partir de este punto empieza a reducirse progresivamente.



(a) Double P-Tree **(b)** Proxy-Tree

Figura 7-8. Influencia de los puertos de servicio en el rendimiento de las arquitecturas P-Tree

Este comportamiento es atribuible a dos causas: la saturación de los puertos de la topología al aumentar el tráfico remoto y a la saturación de los puertos de los usuarios al reducir el número de los mismos disponibles para conectar las redes de usuario (se necesitan 5 puertos para el servidor y 10 puertos para la topología, quedando 9 puertos disponibles para los usuarios).

Con respecto a la arquitectura Proxy-Tree, la mayor capacidad de servicio se logra cuando únicamente se utiliza 3 puertos de servicio, tal y como se puede ver en la figura 7-8b.

7.3.5 Influencia del almacenamiento asignado a los servidores-proxy

En esta sección evaluamos la influencia de la capacidad de almacenamiento de los servidores-proxy sobre el rendimiento de la arquitectura Double P-Tree.

En la figura 7-9, podemos primeramente observar que una capacidad de almacenamiento pequeña implica un bajo rendimiento debido a la gran distancia de servicio requerida para atender las peticiones remotas y los requisitos de ancho de banda de red adicionales.

Con sólo un almacenamiento disponible para el 15% de los videos del catálogo del sistema, el rendimiento de la arquitectura Double P-Tree es equivalente al alcanzado por la arquitectura de servidores independientes, pero requiriendo 6 veces menos almacenamiento.

También, podemos comprobar que el mayor rendimiento se obtiene aproximadamente con una capacidad de almacenamiento alrededor del 25%. En este caso, Double P-Tree mejora al

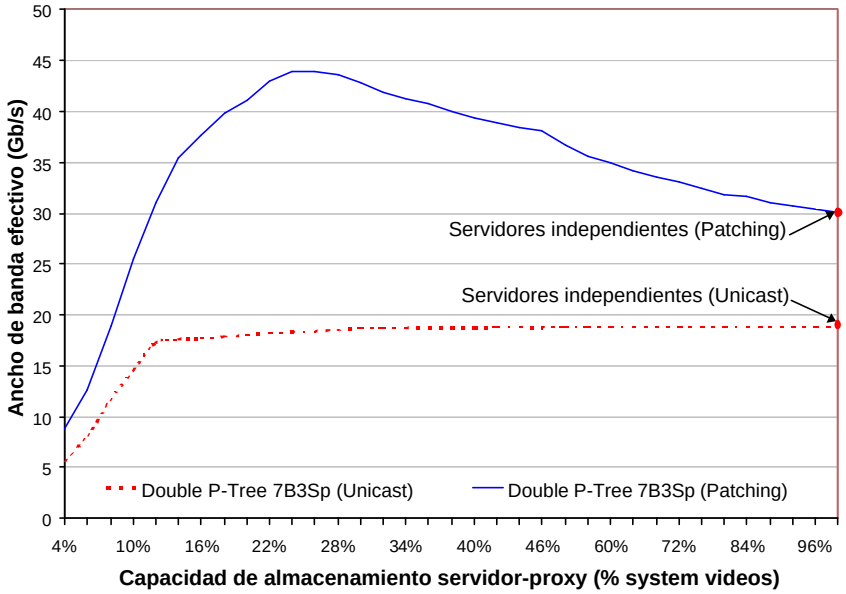


Figura 7-9. Capacidad de servicio de la arquitectura Double P-Tree & capacidad almacenamiento de los servidores-proxy

sistema de servidores independientes en más de un 47% (43.9Gb/s frente a los 30Gb/s). A partir de este punto podemos observar que a medida que la capacidad de crecimiento crece, el rendimiento del sistema decrece hasta alcanzar el rendimiento obtenido por la arquitectura de servidores independientes (en este caso los servidores-proxy tienen suficiente capacidad como para almacenar una copia completa del catálogo de videos).

¿Como es posible que un incremento de los recursos de almacenamiento implique un menor rendimiento? La razón que justifica a este efecto tan interesante puede ser explicado por el hecho que, cuando los servidores-proxy de la arquitectura Double P-Tree tiene una considerable capacidad de almacenamiento (más de un 30%), el comportamiento arquitectural es muy similar al del sistema de servidores independientes. En este caso, los servidores-proxy tienen suficiente almacenamiento para localizar todos los videos como máximo a distancia 2 y, por lo tanto, el resto de almacenamiento se dedica a la cache.

Aumentando el tamaño de la cache se incrementa el número de peticiones atendidas localmente, provocando dos consecuencias. Primero, los puertos de servicio tienen más tráfico, provocando un desbalanceo en el tráfico de las redes locales, y una saturación más rápida de la infraestructura de comunicaciones.

Segundo, existen videos con una popularidad media que, cuando se dispone de una menor capacidad, son gestionados mediante el esquema de mirroring. Los servidores-proxy en donde están asignados estos videos, centralizan todas las peticiones procedentes de los nodos adyacentes, mejorando la probabilidad de compartición de recursos. Si ahora ubicamos estos videos en la cache (replicándolos en todos los servidores-proxy), entonces estamos reduciendo la frecuencia de acceso para cada copia individual del video, reduciendo simultáneamente la probabilidad de compartición (multicast) y el rendimiento del sistema.

El comportamiento que acabamos de describir demuestra la bondad de la utilización del mirroring distribuido (Double P-Tree) frente a distintos mirrors replicados independientes (servidores independientes).

7.4 Resumen

En este apartado vamos a analizar la influencia de cada una de las contribuciones de la tesis sobre el rendimiento final del sistema LVoD. La figura 7-10 muestra la evolución de la capacidad de servicio de las arquitecturas P-Tree a medida que incluimos las distintas optimizaciones.

Inicialmente asumimos la arquitectura Proxy-Tree con 63 redes locales no segmentadas con un ancho de banda de 100 Mb/s y una capacidad en los servidores para el 25% de los

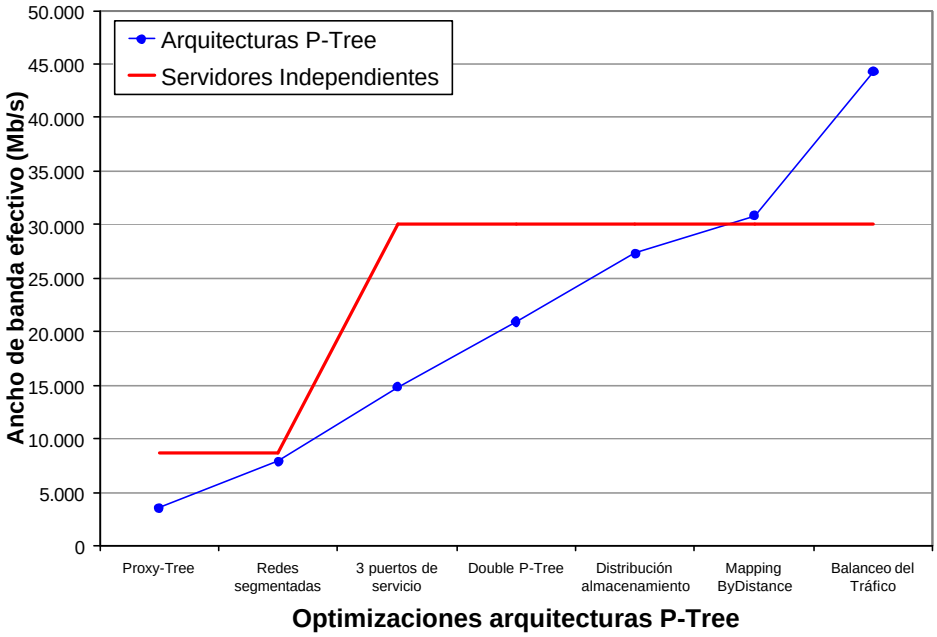


Figura 7-10. Evolución de la capacidad servicio para las diferentes optimizaciones en las arquitecturas P-Tree

contenidos del catálogo del sistema. Este almacenamiento se distribuye en partes iguales entre caching y mirroring. Con esta configuración inicial y utilizando la política de servicio Patching se obtiene un ancho de banda efectivo de 3.5 Gb/s.

En la segunda configuración se utilizan redes segmentadas en las redes locales. Al permitir que el tráfico de cada uno de los enlaces de la topología sea independiente del resto, se logra incrementar el rendimiento del sistema un 125% hasta los 7.9 Gb/s. Este rendimiento supera la capacidad de servicio teorica de 6.3 Gb/s que se obtiene de multiplicar la capacidad de servicio de cada servidor por el número de redes locales (100 Mb/s * 63).

Para aumentar la capacidad de servicio de la arquitectura y poder aprovechar la infrautilización existente en la red de transmisión, en la siguiente configuración se ha aumentado la capacidad de servicio de los servidores-proxy. No obstante, en vez de concentrar todo este tráfico en un único puerto de la red (con el consiguiente aumento de los requisitos de ancho de banda de las redes locales), hemos optado por utilizar 3 puertos de servicio cada uno de ellos con una capacidad de servicio de 100 Mb/s. Mediante esta modificación la capacidad de servicio de la topología se incrementa un 87% adicional, alcanzando los 14.7 Gb/s (a pesar de mantener una distancia media de servicio grande, 2.194).

En la siguiente evolución incorporamos la arquitectura Double P-Tree, utilizando 7 servidores hermanos. Esta arquitectura, gracias a su mayor conectividad, permite reducir la distancia media

de servicio hasta 1.63 y lograr una menor saturación de los enlaces de la topología al distribuir su tráfico (asociado con las peticiones remotas) entre un mayor número de enlaces. La arquitectura Double P-Tree permite mejorar el ancho de banda efectivo del sistema hasta los 20.9 Gb/s (un 41% más).

A continuación incorporamos las técnicas de distribución del almacenamiento para calcular la distribución idónea entre los esquemas de caching y mirroring. Mediante esta técnica pasamos de una distribución equitativa entre los dos esquemas de almacenamiento a una distribución del 56% para el mirror y el 44% para la cache. Esta nueva distribución permite reducir la distancia media de servicio hasta 1.54 y mejorar el rendimiento del sistema en un 30% adicional (hasta los 27.6 Gb/s)

En la penúltima optimización incorporamos en la arquitectura Double P-Tree la política de asignación de videos ByDistance Optimizado. Mediante esta política de mapping se logra una reducción adicional de la distancia media de servicio (1.49) y se alcanza una capacidad de servicio de 30.8 Gb/s (un 13% más). Mediante la inclusión de las políticas de mapping se logra superar el rendimiento de la arquitectura de servidores-independientes.

Por último, incluimos también las políticas de balanceo de tráfico (estáticas y dinámicas) las cuales nos permiten reducir los requisitos de ancho de banda de las redes locales y mejorar el equilibrio de la carga en el sistema. Estas políticas permiten mejorar el rendimiento del sistema en un 44%, alcanzando los 44.3 Gb/s.

Conclusiones y líneas abiertas

Resumen

Este capítulo presenta las conclusiones obtenidas a partir de esta tesis y las líneas futuras de investigación derivadas del presente trabajo.

Conclusiones y principales contribuciones

Nuestro principal objetivo durante el presente trabajo ha sido el diseño de una arquitectura LVoD que cumpla las siguientes premisas: una **escalabilidad** ilimitada, **tolerante a fallos**, que utilice **componentes sencillos** y de **bajo coste** y que permita obtener un sistema **eficiente** con respecto a la compartición de recursos y al **balanceo de la carga** del sistema.

El requisito de una arquitectura que pueda adaptar su tamaño a las necesidades de crecimiento del sistema implica evitar cualquier componente centralizado que se pueda convertir en el cuello de botella para la escalabilidad del sistema. Por lo tanto, nuestro diseño se ha basado en la utilización de arquitecturas totalmente distribuidas.

Para garantizar la tolerancia a fallos de la arquitectura, la gestión del sistema LVoD no puede depender de un único componente, ya que si este elemento falla todo el sistema puede caer con él. Por lo tanto, la utilización de una gestión distribuida del sistema LVoD, unido a una arquitectura distribuida permite lograr un sistema tolerante a fallos.

Para que la arquitectura LVoD también sea tolerante a los fallos que se puedan producir en el sistema de almacenamiento, los contenidos multimedia deben estar replicados y distribuidos en distintos lugares del sistema. De esta forma, si un nodo de almacenamiento cae, existen otras alternativas en donde encontrar los contenidos perdidos.

Para reducir la complejidad de los nodos de servicio y, en general, el coste de los componentes del sistema LvoD, hemos propuesto la utilización de componentes (redes y servidores) sencillos y con pequeños requisitos de prestaciones. Esta estrategia permitirá la utilización de componentes estándar y, por lo tanto, con una mejor relación coste / prestaciones.

Si la eficiencia de un sistema de VoD la definimos como el número de usuarios que se pueden atender con unos recursos acotados (red + nodos de servicio), uno de los puntos clave en el rendimiento de las arquitecturas LVoD es la efectividad de las políticas de compartición de recursos (políticas multicast). La efectividad de las políticas multicast depende del número de usuarios que potencialmente pueden compartir un stream de video, lo cual, en las arquitecturas distribuidas, depende de la conectividad existente entre los nodos de servicio y los usuarios.

Todas estas decisiones de diseño se han plasmado en el diseño de la arquitectura Proxy-Tree. Esta arquitectura se basa en un sistema distribuido compuesto de una topología jerárquica en árbol basada en la utilización de redes locales independientes que facilitan el crecimiento del sistema LVoD. Estas redes utilizan un servidor-proxy local para poder servir las peticiones locales sin necesidad de recurrir al servidor principal y lograr un sistema distribuido.

Para mejorar la escalabilidad del sistema es imprescindible **evitar** los **cuellos de botella** en el diseño de la arquitectura LVoD, por lo tanto, se ha modificado la funcionalidad de los servidores-proxy de forma que puedan gestionar parte de la carga de trabajo que tendría que soportar el servidor principal y su red. En nuestro diseño, los servidores-proxy utilizan dos esquemas de gestión de los contenidos multimedia: el caching que almacena los contenidos más populares y cuyo objetivo es aumentar el volumen de peticiones atendidas localmente; y el mirroring distribuido cuyos objetivos principales son descongestionar el servidor principal (distribuyendo la responsabilidad de gestionar los fallos producidos en las caches locales entre todos los servidores-proxy del sistema) e incrementar la tolerancia a fallos del sistema replicando los contenidos del catálogo. Al distribuir el servicio de las peticiones entre los servidores-proxy, el tamaño del servidor principal requerido (y su coste) es mucho menor que en un sistema parcial ó totalmente centralizado.

Un concepto importante introducido durante este trabajo es el de mirroring distribuido, que permite la gestión conjunta de varios mirrors parciales correspondientes a distintos servidores para almacenar una copia de todos los contenidos multimedia del catálogo. En última instancia, el mirroring distribuido es la pilar fundamental sobre el cual se soporta la gestión distribuida y la escalabilidad de la arquitectura P-Tree.

La arquitectura P-Tree permite adaptar el tamaño del sistema a las necesidades en cada instante de tiempo. La arquitectura propuesta permite tanto la escalabilidad vertical (aumentando la capacidad de servicio del servidor ó del almacenamiento) como la escalabilidad horizontal (incluyendo nuevas redes), alcanzando una gran flexibilidad de crecimiento del sistema LVoD.

Al utilizar una arquitectura distribuida, el sistema resultante es tolerante a fallos. El uso de un sistema de redes locales independientes y la utilización de mirrors distribuidos en los servidores-proxy para replicar las películas del sistema, permite que aunque falle cualquiera de la redes, servidores-proxy ó incluso el servidor principal, se pueda continuar dando servicio parcialmente desde los servidores-proxy de las redes locales adyacentes.

A partir de los resultados proporcionados por el modelo analítico desarrollado, hemos podido *demostrar la escalabilidad ilimitada de la arquitectura Proxy-Tree*. Además, se ha podido constatar que el rendimiento obtenido por esta arquitectura es similar al de las arquitecturas parecidas (un nivel de servidores-proxy) pero sin las limitaciones de crecimiento ó la necesidad de componentes demasiado complejos (servidor y red principal).

El diseño y la evaluación de esta arquitectura han sido publicados en:

[Cor01] F. Cores, A. Ripoll, E. Luque, "A fully scalable and distributed architecture for video-on-demand", PROMS'01, LNCS 2213, pp 59-73, Twente, Holland, Oct 2001.

A partir del análisis de los resultados de la arquitectura P-Tree, hemos podido observar que las arquitecturas distribuidas de VoD que no tienen todos los contenidos replicados, tienen que sacrificar una parte de su eficiencia para soportar la tolerancia a fallos y la escalabilidad. Esta reducción en el rendimiento del sistema viene definido por el aumento de recursos de ancho de banda de red que requieren las peticiones que no pueden ser atendidas por el servidor-proxy local y que tienen que ser gestionadas por servidores remotos.

Un concepto clave para evaluar el ancho de banda de red adicional requerido por los sistemas de VoD distribuidos es la **distancia media de servicio**. Esta distancia refleja el número promedio de redes que deben cruzar los streams de video en su transmisión desde el servidor hasta el usuario que realizo la petición. La distancia media de servicio depende del almacenamiento disponible en cada uno de los nodos de servicio, de los esquemas bajo los cuales se gestiona este almacenamiento y sobre todo de la conectividad de la topología subyacente en la arquitectura del sistema VoD.

Para reducir la penalización asociada con la distribución de la gestión y los contenidos en la arquitectura Proxy-Tree, se ha analizado en profundidad el comportamiento de esta arquitectura, llegando a una serie de conclusiones interesantes:

- *El incremento de la eficiencia de la arquitectura P-Tree sin aumentar los recursos de la arquitectura, solo se puede lograr incrementado la conectividad de la topología*. Una mayor conectividad entre las servidores-proxy, permite lograr dos objetivos: aumentar la capacidad de los mirrors distribuidos (formados por los mirrors parciales situados a una distancia de una determinada red local), y aumentar el porcentaje de almacenamiento que se puede dedicar al esquema de caching ya que el esquema de mirror disminuye sus requerimientos. Ambos objetivos nos van a permitir reducir la distancia media de servicio de la arquitectura.
- Utilizar un mayor orden en el árbol de la topología mejora la conectividad localmente en algunos nodos pero la empeora en otros, obteniendo al final la misma conectividad media global. Por lo tanto, *el incremento del orden del árbol de la topología P-Tree no es una solución factible para mejorar su rendimiento.*

- *La interconexión de las redes locales* de un mismo nivel (redes / servidores hermanos) nos *permite incrementar la conectividad de la topología* de forma sencilla y flexible, a costa de reducir el número de puertos disponibles en los switches. Esta nuevas conexiones introducen el concepto de servidores-proxy hermanos, que son aquellos servidores de un mismo nivel, conectados entre si a través de enlaces.
- Una de las diferencias existentes entre una arquitectura VoD centralizada y una arquitectura distribuida es la distribución del tráfico en las redes. En la arquitectura centralizada el tráfico esta centralizado en un puerto (puerto de servicio) que es el que conecta el servidor central con la red central. Sin embargo en las arquitecturas totalmente distribuidas, como la arquitectura P-Tree, el tráfico esta más distribuido entre distintos fuentes ó puertos (enlaces de la topología). Nosotros podemos aprovechar esta característica utilizando redes segmentadas en las redes locales que permitan reducir los requisitos de red de las arquitecturas distribuidas.
- Al utilizar redes segmentadas, la *conexión de los servidores* a la red de transmisión *utilizando diversos puertos* (puertos de servicio), *permite reducir los requisitos de red* asociados con un alto volumen de tráfico centralizado en un único puerto.

A partir de estas conclusiones, hemos diseñado una nueva arquitectura distribuida para LVoD denominada Double P-Tree. Esta arquitectura se basa en el incremento de la conectividad de la topología mediante la utilización del concepto de servidores hermanos y la utilización de redes segmentadas, logrando un sistema LVoD completamente distribuido al eliminar el servidor principal y la red principal, distribuyendo su funcionalidad entre todos los servidores-proxy locales.

Gracias a estas modificaciones, la arquitectura Double P-Tree ha alcanzado una mayor eficiencia. *Esta arquitectura mejora el rendimiento de la arquitectura P-Tree en un 210% y supera al sistema de un nivel de servidores-proxy en un 350%.* Double P-Tree también ha demostrado ser capaz de alcanzar una *efectividad similar a la arquitectura de servidores independientes*, con unos requisitos de *almacenamiento 5 veces menor*. La arquitectura Double P-Tree también permite mejorar la tolerancia a fallos.

Otra de las conclusiones destacables que podemos extraer de la arquitectura Double P-Tree es la *independencia de los conceptos de diseño fundamentales de las arquitecturas P-Tree* (redes independientes, servidores-proxy, mirroring, caching, gestión distribuida, distribución de la carga y el tráfico) *de la topología utilizada para la interconexión de las distintos componentes*.

El análisis del rendimiento de la arquitectura P-Tree, así como el diseño y evaluación de la arquitectura Double P-Tree han sido publicados en los siguientes artículos:

[Cor02] F. Cores, A. Ripoll, E. Luque, "Double P-Tree: A Distributed Architecture for Large-Scale Video-on-Demand", Euro-Par 2002, LNCS 2400, pp. 816-825, Aug. 2002.

[Cor02b] F. Cores, A. Ripoll, E. Luque, "A Cost-effective Distributed Architecture for Large-Scale Video-on-Demand", Procedings of IEEE Int. Conference on Multimedia and Expo (ICME), pp 13-16, 2002.

La correcta gestión del almacenamiento de los servidores-proxy que conforman un sistema distribuido de VoD es de vital importancia para el rendimiento del sistema. En concreto, se ha demostrado que *la distancia media de servicio se ve afectada* por dos aspectos relacionados con el almacenamiento: el porcentaje de *almacenamiento* de los servidores-proxy *dedicado a cada uno de los esquemas de gestión de contenidos* (caching / mirroring) y *la asignación de los contenidos multimedia a los mirrors* de los servidores-proxy del sistema.

Los resultados obtenidos han demostrado que la utilización en exclusiva del esquema de caching (tal y como proponen las arquitecturas basadas en servidores-proxy de un nivel) ó solo el esquema de mirroring (tal y como proponen las arquitecturas de servidores independientes) no son una buena solución por si solos. Las arquitecturas distribuidas de LVoD (que no repliquen todos los contenido en todos los nodos) exigen la utilización combinada de ambos esquemas (caching y mirroring distribuido) para optimizar la capacidad de servicio del sistema.

Entre los dos esquemas de gestión, el esquema de mirroring es el que tiene una mayor preponderancia. La razón del mayor porcentaje de almacenamiento asignado al mirror estriba en la alta replicación asociada con el esquema de caching. Por lo tanto, solo es recomendable la utilización de la cache para almacenar contenidos con una alta popularidad (sobre todo cuando la capacidad de los servidores-proxy es escasa) que permitan incrementar el porcentaje de peticiones servidas localmente, y sin afectar a la eficiencia de los mirrors distribuidos.

La distribución de los contenidos en los mirrors de los servidores y el orden en que se realiza las asignaciones, tiene una importante influencia sobre la distancia media de servicio de las arquitecturas P-Tree, y por lo tanto, afecta a la capacidad de servicio final del sistema LVoD.

Se ha propuesto una política de mapping, denominada ByDistance, basada en una heurística para reducir la distancia media de servicio, que ha demostrado su eficiencia en el incremento del rendimiento de las arquitecturas P-Tree. Esta política logra incrementar la capacidad de servicio de la arquitectura Double P-Tree en un 65% respecto a políticas más sencillas. Se han analizado distintos órdenes a la hora de procesar los servidores-proxy durante la asignación de contenidos, comprobando su influencia en la eficiencia de las políticas de mapping. Se ha observado una diferencia de rendimiento entre los distintos órdenes de procesamiento de un 15%.

Otro factor importante en el rendimiento de las arquitecturas distribuidas es el balanceo de la carga y del tráfico recibido por los distintos componentes de la arquitectura distribuida.

El **desbalanceo global** entre la carga y el tráfico recibido se puede atribuir principalmente a la diferente conectividad de los servidores del sistema, la distribución no uniforme de las peticiones de los usuarios y / ó a la distribución de los contenidos entre los distintos mirrors de los servidores-proxy.

El **desbalanceo local** procede de la distribución del tráfico de cada red local entre los distintos puertos de la misma. Debido a las características de las redes segmentadas, *el desbalanceo entre el tráfico de los puertos puede provocar una reducción en la eficiencia de las arquitecturas distribuidas*, incrementando los requisitos de ancho de banda de red y disminuyendo la capacidad de servicio del sistema.

El desbalanceo local en los puertos de las redes se puede atribuir al desequilibrio existente entre los distintos tipos de puertos: puertos de servicio, puertos de la topología y puertos de los usuarios. El tráfico asociado con los puertos de servicio y puertos de usuarios no puede ser redistribuido sin modificar la arquitectura. Por lo tanto, para poder reducir el desbalanceo de las redes locales y con ellos los requisitos de ancho de banda de red del sistema, la solución más factible consiste en equilibrar el tráfico asociado con los puertos de la topología.

La distribución del tráfico entre los puertos de la topología depende en última instancia de la distribución de las peticiones que no se pueden servir localmente entre los distintos servidores remotos. La distribución de estas peticiones depende a su vez de como se hayan distribuido los contenidos entre los distintos servidores-proxy.

Se han propuesto dos tipos de políticas diferentes para evitar el desbalanceo en las redes locales: políticas de balanceo estáticas (mapping ByDistance Gradual) y políticas de balanceo dinámicas (balanceo de peticiones). Ambas políticas han demostrado una gran compatibilidad, logrando mediante la utilización combinada de ambas políticas que el desbalanceo del tráfico en los puertos de la topología sea únicamente del 4% (en contraposición al 20% de desbalanceo cuando no se utilizan dichas políticas). Este porcentaje de desbalanceo se puede considerar

residual. Mediante la consecución de un tráfico balanceado en las redes locales, se logra incrementar la capacidad de servicio de las arquitecturas Double P-Tree en un 25%.

Estas optimizaciones realizadas sobre las arquitecturas P-Tree han sido publicadas en los siguientes artículos:

[Cor03] F.Cores, A.Ripoll, B.Qazzaz, R.Suppi, X.Yang, P.Hernandez, E.Luque, "Exploiting Traffic Balancing and Multicast Efficiency in Distributed Video-on-Demand Architectures", Euro-Par 2003, LNCS 2790, pp. 859-869, Aug. 2003.

[Cor03b] F.Cores, A. Ripoll, X.Y. Yang, B. Qazzaz, R. Suppi, P. Hernández, E. Luque, "Improving Bandwidth Efficiency in Distributed Video-on-Demand Architectures", Parallel Procesing Letters, 2003. Articulo invitado (en prensa).

Para realizar la evaluación de las distintas políticas de gestión de las arquitecturas distribuidas (políticas multicast, de gestión del almacenamiento, de asignación de videos balanceo del tráfico, etc...) se requiere de la utilización de herramientas de evaluación más potentes y flexibles que los modelos analíticos utilizados durante la primera fase de este trabajo de investigación.

Sin embargo, dichas herramientas hoy en día no están disponibles. Por lo tanto, uno de los trabajos importantes realizados durante la tesis ha consistido en el diseño e implementación de un **entorno de simulación**, denominado ***VoDSim***, que permita evaluar las distintas arquitecturas LVoD estudiadas, e incorporar las diferentes optimizaciones propuestas durante estas investigación.

Durante el diseño se ha buscado dotar a la herramienta de una gran flexibilidad que facilite su ampliación y la incorporación de nuevas características, utilizar diferentes niveles de abstracción en el modelo, una gran versatilidad en la configuración de los parámetros de simulación y disponer de un amplio abanico de métricas de rendimiento del sistema. Los resultados de simulación han sido validados frente a los distintos modelos analíticos presentados, mostrando una elevada correlación.

Mediante la utilización del entorno de simulación se han podido evaluar las distintas políticas propuestas y realizar un análisis comparativo de la eficiencia de las distintas arquitecturas distribuidas.

Estos resultados muestran que cuando se utilizan políticas unicast la arquitectura Double P-Tree logra resultados similares ó mejores que el resto de arquitecturas distribuidas (requiriendo en la mayoría de los casos menos recursos). Asumiendo un escenario más realista, y por lo tanto utilizando políticas de servicio multicast, la arquitectura Double P-Tree con 7 hermanos se muestra como la mejor opción, mejorando en más de un 47% la capacidad de servicio de la arquitectura de servidores independientes (utilizando 4 veces menos recursos de almacenamiento), en un 190% el rendimiento de la arquitectura de servidores-proxy de un nivel y en un 194% el rendimiento de la arquitectura P-Tree (44.3Gb/s con respecto a 15.1Gb/s).

Finalmente, se ha mostrado que una completa replicación de los contenidos del catálogo en cada uno de los nodos de servicio (servidores independientes) no solo requiere más recursos de almacenamiento, sino que obtiene un peor rendimiento en comparación con la utilización del mirroring distribuido (Double P-Tree).

Líneas abiertas

De la experiencia obtenida durante la realización de este trabajo han surgido nuevos retos y líneas de trabajo futuros, que a continuación enumeramos:

- Análisis de la posible utilización de otras topologías dentro de las arquitecturas P-Tree.

  Este estudio se basaría en la utilización de topologías en hipercubo como las utilizadas por la arquitectura Ncube [NCub1] [NCub2], pero conservando los conceptos propios de las arquitecturas distribuidas P-Tree: utilización de redes independientes segmentadas y servidores-proxy, gestión distribuida, utilización de esquemas de gestión del almacenamiento de caching y mirroring distribuido.

- Adaptación de las arquitecturas P-Tree a un entorno heterogéneo (desde el punto de vista de la topología, los recursos y los protocolos de comunicaciones utilizados).

  Una de las líneas de investigación futura más interesante es la adaptación de la arquitecturas P-Tree a entornos de interconexión heterogéneos, como Internet.

  Este paso, implica la modificación de las políticas de gestión de la arquitectura de forma que se adapten a los distintos grados de conectividad, los distintos tipos de red y anchos de banda y la utilización compartida de los recursos de transmisión.

- Incrementar el nivel de distribución de la arquitectura mediante la utilización de políticas peer-to-peer ó chaining.

  Mediante la utilización de políticas *peer-to-peer* ó de *chaining* se puede profundizar en el nivel de distribución de la arquitecturas P-Tree. Estas técnicas pueden facilitar el diseño de sistemas DVoD que requieran un reducido ancho de banda de servicio en los servidores del sistema (la mayor parte de la responsabilidad del servicio a los usuarios se delega a los STB de los propios usuarios).

- Proponer políticas de balanceo global de la carga de los servidores y de las redes para reducir los requisitos del sistema.

  Otra futura línea de investigación está orientada a analizar y tratar de resolver el desbalanceo global que se puede producir en las arquitecturas distribuidas, mediante la modificación de la política de balanceo de peticiones.

  Para afrontar este problema, la política de balanceo de peticiones, también debería tener en cuenta el desequilibrio de la carga entre los servidores-proxy a la hora de tomar la decisión de quién tiene que servir una petición remota.

  La asignación y la distribución del almacenamiento en los servidores-proxy se debería realizar teniendo en cuenta su conectividad, con el objetivo de reducir los desequilibrios de carga provocados por la distinta conectividad de los distintos niveles de las topologías P-Tree.

- Mejora de las políticas de balanceo de tráfico en las redes.

  Como línea de futura investigación dentro de este ámbito proponemos el equilibrio del tráfico de los restantes puertos de las redes (servicio y usuarios) mediante políticas que permitan desviar el tráfico de los puertos de servicio (normalmente los más saturados) a los puertos de usuario.

  Este objetivo se puede lograr incrementado la distribución de la arquitectura, permitiendo que los mismos usuarios puedan servir peticiones a otros usuarios mediante la utilización de políticas de *chaining* y por lo tanto, transfiriendo el tráfico de los puertos de servicio hacia los puertos de los usuarios.

- Analizar políticas de distribución de los contenidos en el almacenamiento de los servidores-proxy que tengan en cuenta el ciclo de vida de los contenidos y su popularidad actual.

  La asignación de contenidos a los mirrors actuales se realiza de forma estática, y atendiendo exclusivamente a criterios de reducción de la distancia media de servicio. Estas

políticas han demostrado su efectividad cuando los servidores-proxy tienen suficiente almacenamiento como para almacenar todos los contenidos dentro del mirror distribuido situado a distancia 2 de cualquier servidor.

Para poder adaptar los contenidos de los mirrors de forma dinámica a la popularidad variable de los videos se propone la inclusión de un tercer esquema de gestión ya introducido a lo largo de este trabajo: el caching cooperativo. Mediante este esquema de gestión de contenidos intermedio entre la cache autónoma y el mirroring distribuido se propone la realización de políticas que permitan que una parte de los contenidos de los servidores-proxy se adapte dinámicamente en función de la popularidad. Esta política también permitirá lograr que el número de copias de cada contenido en el sistema sea proporcional a su popularidad.

Referencias

[Bak02] Bakiras, S.; Li, V.O.K.; “Maximizing the number of users in an interactive video-on-demand system”, IEEE Transactions on Broadcasting , Volume: 48 Issue: 4, Page(s): 281-292 , Dec. 2002.

[Bra97] R. Braden, L. Zhang, S. Berson, S. Herzog, S. Jamin, “Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification”, ftp://ftp.isi.edu/in-notes/rfc2205.txt

[Bra99] Branch, P., Egan, G., Tonkin, B., "Modeling Interactive Behaviour of a Video Based Multimedia System", IEEE International Conference on Communications, Vancouver, June 1999.

[Bra00] M. K. Bradshaw, B. Wang, S. Sen, L. Gao, D. Towsley, “Periodic Broadcast and Patching Services -Implementation, Measurement, and Analysis in an Internet Streaming Video Testbed“, UMass Computer Science Technical Report 2000-56.

[Ber96] C. Bernhardt and E. Biersack, “The server array: A scalable video server architecture”, High-Speed Networks for Multimedia Applicatiosn. Norwell, MA: Kluwer, 1996.

[Cai99] Y. Cai, K. Hua, and K. Vu, "Optimizing patching performance", in Proc. SPIE/ACM Conference on Multimedia Computing and Networking, January 1999.

[Cha99] S.-H. G. Chan and F. Tobagi, “Caching schemes for distributed video services”, in Proceedings of the 1999 IEEE International Conference on Communications (ICC'99), June 1999.

[Chu00] J.C.I. Chuang, M.A. Sirbu, “Distributed Network Storage with Quality-of-Service Guarantees”, Journal of Network and Computer Applications 23(3), pp. 163-185, July 2000.

[Cisc1] Inc Cisco Systems, “Internetworking Design Guide”, Cisco online book, http://www.cisco.com/univercd/cc/td/doc/cisintwk/idg4/

[Cisc00] Inc Cisco Systems, “Cisco CCIE Fundamentals: Network Design and Case Studies”, Cisco Press, ISBN: 1578701678, 2000.

[Cor01] F. Cores, A. Ripoll, E. Luque, “A fully scalable and distributed architecture for video-on-demand”, PROMS’01, LNCS 2213, pp 59-73, Twente, Holland, Oct 2001

[Cor02] F. Cores, A. Ripoll, E. Luque, “Double P-Tree: A Distributed Architecture for Large-Scale Video-on-Demand”, Euro-Par 2002, LNCS 2400, pp. 816-825, Aug. 2002.

[Cor02b] F. Cores, A. Ripoll, E. Luque, “A Cost-effective Distributed Architecture for Large-Scale Video-on-Demand”, Procedings of IEEE Int. Conference on Multimedia and Expo (ICME), pp 13-16, 2002.

[Cor03] F.Cores, A.Ripoll, B.Qazzaz, R.Suppi, X.Yang, P.Hernandez, E.Luque, “Exploiting Traffic Balancing and Multicast Efficiency in Distributed Video-on-Demand Architectures”, Euro-Par 2003, LNCS 2790, pp. 859-869, Aug. 2003.

[Cor03b] F.Cores, A. Ripoll, X.Y. Yang, B. Qazzaz, R. Suppi, P. Hernández, E. Luque, “Improving Bandwidth Efficiency in Distributed Video-on-Demand Architectures”, Parallel Procesing Letters, 2003, Pendiente de publicación.

[Dan93] A. Dan and D. Sitaram, "Buffer Management Policy for an On-Demand Video Server", RC 19347, IBM Research Division, 1993.

[Dan94] A. Dan, D. Sitaram, P. Shahabuddin, “Scheduling Policies for an On-Demand Video Server with Batching”, Proc. 2nd ACM Int’l. Multimedia Conference (ACM MULTIMEDIA ’94), San Francisco, CA, Oct. 1994, pp. 15-23.

[Dan96] A. Dan, D. Sitaram, P. Shahabuddin, "Dynamic batching policies for an on-demand video server" Multimedia Systems 4, pp. 112-121, June 1996.

[Duc02] Duc A. Tran, Kien A. Hua, and Tai T. Do. Scalable Media Streaming in Large P2P Networks. In Proceedings of ACM Multimedia Conference (SIGMM 2002), pp. 247-250, December 1-6, 2002.

[Dri94] Dring K.-H., “Extension of public networks for VoD services”, Deutche Telekom FTZ, 1994.

[Eag99] D. L. Eager, M. K. Vernon, and J. Zahorjan, "Minimizing Bandwidth Requirements for On-Demand Data Delivery", Proc. MIS'99, Indian Wells, CA, Oct. 1999.

[Eag00] D. L. Eager, M. K. Vernon, and J. Zahorjan, "Bandwidth Skimming: A Technique for Cost-Effective Video-on-Demand", Proc. IS&T/SPIE Conf. on Multimedia Computing and Networking (MMCN 2000), San Jose, CA, January 2000.

[Fen95] W. Feng and S. Sechrest, “Smoothing and buffering for delivery of prerecorded compressed video,” Comput. Commun., vol. 18, pp. 709–717, Oct. 1995.

[Fri99] J.Fritz, “Caught up on Video”, Data Communications, octubre 1999, pp. 51-55.

[Gol96] L. Golubchic, J. C. S. Liu, and R. R. Muntz. “Adaptive piggybacking: a novel technique for data sharing in video-on-demand storage servers”, ACM Multimedia Systems Journal, Vol. 4, No. 3, 140-155, 1996.

[Gol01] L. Golubchik, R. Muntz, Cheng-Fu Chou, S. Berson, “Design of fault-tolerant large-scale VOD servers: With emphasis on high-performance and low-cost”, Parallel and Distributed Systems, IEEE Transactions on , Volume: 12 Issue: 4 , April 2001, Page(s): 363 –386.

[Gri97] C. Griwodz, Michael Bär, and Lars C. Wolf, “Long-term Movie Popularity in Video-on-Demand Systems”, In Proc. of ACM Multimedia'97, pages 349-357, November 1997.

[Gri00] C. Griwodz “Wide-area True Video-on-Demand by a Decentralized Cache-based Distribution Infrastructure”, PhD thesis, Darmstadt University of Technology, April 2000.

[Hof99] M. Hofmann, T. S. E. Ng, K. Guo, S. Paul, and H. Zhang, "Caching Techniques for Streaming Multimedia over the Internet," Bell Laboratories Technical Report, Abril 1999.

[Hua97] K. A. Hua, S. Sheu, "Skyscraper Broadcasting: a new broadcasting scheme for metropolitan video-on-demand systems," in SIGCOMM 97, pp. 89--100, ACM, (Cannes, France), Sept. 1997.

[Hua98] K. A. Hua, Ying Cai and Simon Sheu, “Patching: A Multicast Tecnique for true Video-on-Demand Services”, ACM Multimedia 1998, pages 191-200.

[Hwa01] Ren-Hung Hwang and Pin-Hao Chi, "Fast Optimal Video Placement Algorithms for Hierar-chical Video-on-Demand Systems," IEEE Transactions on Broadcasting, Vol. 47, Issue:4, pp. 357–366, Diciembre 2001.

[Hwa02] Eenjun Hwang; Prabhakaran, B.; Subrahmanian, V.S.; “Presentation planning for distributed VoD systems“, IEEE Transactions on Knowledge and Data Engineering , Volume: 14 Issue: 5 , Pages: 1059 –1077, Sept.-Oct. 2002

[Juh97] Li-Shen Juhn; Li-Ming Tseng; “Harmonic broadcasting for video-on-demand service Broadcasting”, IEEE Transactions on , Volume: 43 Issue: 3 , Page(s): 268 -271, Sept. 1997

[Kas1] “Cost-Effective Video Delivery for Large-Scale VOD Solutions”, Technical White Paper, disponible online en: http://www.kasenna.com/newkasenna/products/resources.jsp

[Kas2] “Kasenna MediaBase: A Scalable Software Platform for Video Management, Distribution and Delivery”, Technical White Paper, http://www.kasenna.com/newkasenna/products/resources.jsp

[Kun01] Kun-Lung W., Philip S. Y. and Joel L. W., “Segment-based proxy caching of multimedia streams”, In Proc. of the loth International WWW Conference, Hong Kong, 2001.

[Lau98] S.W. Lau, J.C.S. Lui and L. Golubchik, “Merging Video Streams in a Multimedia Storage Server: Complexity and Heuristics”, Multimedia Systems, 6(1), 1998, 29-42.

[Lee98] J.Y.B. Lee, “Parallel Video Servers: A Tutorial”, IEEE Multimedia, vol.5(2), April-June 1998, pp. 20-28.

[Lee02] J.Y.B. Lee, “Staggered push - a linearly scalable architecture for push-based parallel video servers”, IEEE Transactions on Multimedia, Volume: 4 Issue: 4, pp 423 -433, Dec. 2002.

[Leu02] W. T. Leung and Jack Y. B. Lee, "A Server-less Architecture for Building Scalable, Reliable, and Cost-Effective Video-on-demand Systems," Internet2 Workshop on Collaborative Computing in Higher Education: Peer-to-Peer and Beyond, Tempe, AZ, USA, 30-31 Jan 2002.

[NCub1] “n4x”, Technical Paper, http://www.ncube.com/vod/index.html

[NCub2] “Integrated VOD Systems”, Technical Paper, online en: http://www.ncube.com/vod/index.html

[Net1] “Network Design Guidelines”, Extreme networks withepaper, http://www.extremenetworks.com/libraries/whitepapers/technology/design.asp

[Ouy98] Ouveysi I., Ka-Cheong Wong; Chan, S.; Ko, K.T., “Video placement and dynamic routing algorithms for video-on-demand networks”, IEEE Global Telecommunications Conference, vol.2 , pp. 658–663, Nov. 1998.

[Pad02] V.N. Padmanabhan, H.J. Wang, P.A. Chou, “Distributing Streaming Media Content Using Cooperative Networking”, Technical Report MSR-TR-2002-37, Microsoft Research, 2002.

[Pai94] PairGain, "Enhancing the Performance and Application of Copper Cable with HDSL". A Technology Brief of PairGain Technologies, Inc. http://www.telechoid.com/xdslnewz.

[Par99] J.-F. Paris, “A simple low-bandwidth broadcasting protocol,” in International Conference on Computer Communication and Network, 1999, pp. 118–123.

[Pat88] D. Patterson, G. Gibson and R. Katz, “A Case for Redundant Array of Inexpensive Disks (RAID)”, ACM SIGMOD’88, pp. 109-116, Junio 1988.

[Pin00] Chor Ping Low; Hongtao Yu; Ng, J.M.; Qingping Lin; Atif, Y.; “An efficient algorithm for the video server selection problem” IEEE Global Telecommunications Conference, GLOBECOM '00, Volume: 3, Pages: 1329 -1333, Dec. 2000.

[Qaz03a] B. Qazzaz, R. Suppi, F. Cores, A. Ripoll, X. Yang, P. Hernandez, E. Luque. “Admission Control Policies For Video On Demand Brokers” ICME Multimedia & Expo Proceedings, Baltimore-Myryland, USA, July 2003.

[Qaz03b] B. Qazzaz, R. Suppi, F. Cores, A. Ripoll, P. Hernandez, E. Luque, “Providing Interactive Video on Demand Services in Distributed Architecture”, Proceedings of 29th EUROMICRO Conference, Belek-Antalya, Turkey, September 2003.

[Rei98] Reisslein, M.; Ross, K.W.; “High-performance prefetching protocols for VBR prerecorded video”, Network, IEEE , Volume: 12 Issue: 6 , Page(s): 46 –55, November/December 1998.

[Rex99] J. Rexford and D. Towsley, “Smoothing variable-bit-rate video in an internetwork,” IEEE/ACM Trans. Networking, vol. 7, pp. 202–215, Apr. 1999.

[Riz98] L. Rizzo and Lm Vicisano, “Replacement policies for a proxy cache.” Technical Report RN/98/13, UCL-CS, 1998.

[Sch95] F.Schaffa, Jean-Paul Nussbaumer, ”On Bandwidth and Storage Tradeoffs in Multimedia Distribution Networks”, in Proc. lnfocom, 1995, pp. 1020-1026

[Sch96] H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson, ”RTP: A Transport Protocol for Real-Time Applications”, ftp://ftp.isi.edu/in-notes/rfc1889.txt

[Sen99] S. Sen, L. Gao, J. Rexford, and D. Towsley, "Optimal Patching Schemes for Efficient Multimedia Streaming ", Proc. 9 th Int'l Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV'99), Basking Ridge, NJ, June 1999.

[Sen99b] S. Sen, L. Gao, J. Rexford, and D. Towsley, “Proxy prefix caching for multimedia streams” In Proceedings of the IEEE INFOCOM, 1999.

[Ser00] D. N. Serpanos, A. Bouloutas, “Centralized versus distributed multimedia servers”, IEEE Transactions on Circuits and Systems for Video Technology, vol.10(8), Dec. 2000, pp.1438-1449.

[Sha02] C.Shahabi, F.Banei-Kashani, “Decentralized Resource Management for a Distributed Continuous Media Server“, IEEE Transaction on Parallel and Distributed Systems, vol 13 (5), Julio 2002.

[Sha02b] C.Shahabi, R.Zimmermann, K.Fu, S.Y Didi Yao, “Yima: A Second-Generation Continuous Media Server”, Computer, Junio 2002.

[She97] S. Sheu, K. A. Hua, and W. Tavanapong “Chaining: A Generalized Batching Technique for Video-On-Demand Systems”, In Proc. IEEE Int’l Conf. On Multimedia Compunting and Systems (ICMCS)’97, Ottawa, Ontario, Canada, June 3-6, 1997, pp. 110-117.

[Sim1] “SimPack Simulation ToolKit”, http://www.cisi.ufl.edu/fishwick/simpack.html

[Sit00] D. Sitaram, A. Dan, "Multimedia Servers: applications, environments, and design", Morgan Kaufmann Publishers, ISBN 1-55860-430-8, 2000.

[Ste95] Alexander Stepanov, “The Standard Template Library”, Silicon Graphics Inc, 1995.

[Stl03] “Standard Template Library Programmer's Guide”, SGI, http://www.sgi.com/tech/stl/

[Sup00] R. Suppi, F. Cores, E. Luque: “Improving Optimistic PDES in PVM Environments. “, PVM/MPI 2000, LNCS 1908: pp. 304-312.

[Sup00b] R. Suppi, F. Cores, E. Luque, An Efficient Method for Improving Large Optimistic PDES.”, MASCOTS 2000, pp 351-357.

[Sup01] R. Suppi, F. Cores, E. Luque, ”PDES: A Case Study Using the Switch Time Warp. “, PVM/MPI 2001, LNCS 2131, pp. 327-334.

[Swt1] “Multilayer Switching Reference Guide”, http://www.bcr.com/bcrmag/mlsreference.asp

[Teo72] T.Teory and T.B.Pinkerton, “A Comparative Analysis of Disk Scheduling Policies”, Communications of the ACM, 15(3):177-184, 1972.

[Tew96] Tewari, R.; Mukherjee, R.; Dias, D.M.; Vin, H.M., “Design and performance tradeoffs in clustered video servers”, Proceedings of the Third IEEE International Conference on Multimedia Computing and Systems, pp 144 –150, June 1996

[Tob93] F.A. Tobagi, J. Pang, R. Baird, and M. Gang, “Streaming RAID–A Disk Array Management System For Video Files”, Proc. ACM Multimedia Conf., pp. 393-399, 1993.

[Tob95] F. A. Tobagi, “Distance learning with digital video,” IEEE Multimedia Magazine, pp. 90-94, Spring 1995.

[Ven96] N. Venkatasubramanian and S. Ramanathan, “Effective load management for scalable video servers”, Technical report, Hewlett-Packard Research Laboratories, Feb 1996.

[Ven97] N.Venkatasubramanian and S.Ramanathan, “Load Management in Distributed Video Servers”, Procs. of International Conference on Distributed Computing Systems (ICDCS'97), 1997, pp.31-39.

[Vis96] S. Viswanathan and T. Imielinski, "Metropolitan area video-on-demand service using pyramid broadcasting," Multimedia Systems 4, pp. 197--208, Aug. 1996.

[Wan99] J. Wang. “A survey of web caching schemes for the internet”, ACM Computer and Communication. Review, pages 36--46, 1999

[Wei96] Johannes Weidl, “The Standard Template Library Tutorial”, Information Systems Institute, Distributed Systems Department Technical University Vienna, 1996

[Yu92] P.S Yu, M.S. Chen, and D.D. Kandlur, “Grouped Sweeping Scheduling for Multimedia Storage Management”, Proc. Of Third International Workshop on Network and Operating System Support for Digital Audio and Video, San Diego, pp 38-49, November 1992.