Capítulo 2

Proxy-Tree una Arquitectura Escalable para LVoD

Resumen

En este capítulo se propone un arquitectura LVoD escalable denominada Proxy-Tree, basada en un sistema jerárquico de servidores-proxy interconectados mediante una topología en árbol. A continuación se especifican las políticas de gestión de los contenidos (mirroring y caching) que permiten distribuir la gestión del sistema entre todos los componentes del mismo. Por último, se desarrolla un modelo analítico para esta arquitectura, mediante el cual se evaluará las prestaciones de la arquitectura Proxy-Tree y se demostrará su escalabilidad.

2.1 Introducción

En el presente capítulo vamos a proponer una nueva arquitectura escalable para sistemas de LVoD. Los objetivos de diseño de la arquitectura se derivan del análisis realizado en el apartado 1.5 del capítulo anterior.

En la tabla 2-1 resumimos las principales características que debería poseer una arquitectura LVoD ideal. De las distintas características, las más importantes sin lugar a dudas son las referentes a la escalabilidad, capacidad de servicio y coste final del sistema.

Tabla 2-1. Características ideales para las arquitecturas LVoD

Arquitectura	Alta capacidad de servicio	Grandes anchos de banda	Tolerante a fallos	Escalable	Compartición de recursos	Coste
Ideal	**Si**	**No**	**Si**	**Si**	**Si**	**Razonable**

Para alcanzar estos requerimientos proponemos diseñar una arquitectura distribuida en forma de árbol, con una escalabilidad ilimitada y basada en servidores con menores requisitos de servicio y, por lo tanto, menos complejos, y redes que no requieran grandes anchos de banda. Al tratarse de un sistema orientado al publico, la tolerancia a fallos también es necesaria y también la deberemos tener en cuenta a la hora de diseñar el sistema.

El sistema LVoD resultante debe tener un coste razonable, por lo tanto es imprescindible reducir al máximo el tamaño inicial del sistema (creciendo a medida que las necesidades lo requieran), y evitar la utilización de componentes demasiado costosas.

Estas características nos definen una serie de objetivos que debemos cumplir a la hora de diseñar una arquitectura de VoD a gran escala:

1. Alta capacidad de servicio de videos, independiente de la tecnología disponible y adaptable de forma flexible a las necesidades del sistema.
2. Escalabilidad ilimitada con unos costes acotados.
3. Tolerancia a fallos.
4. Capacidad de servicio.

Para cumplir estos objetivos se propone una arquitectura distribuida que hemos denominado Proxy-Tree, la cual se presentará en el siguiente apartado.

2.2 Arquitectura propuesta: Proxy-Tree

La mayoría de los objetivos planteados sobre nuestra arquitectura dependen de la capacidad de crecimiento del sistema. Por lo tanto, nuestro diseño de la arquitectura LVoD se orientará en una primera instancia a desarrollar la capacidad de crecimiento del sistema.

2.2.1 Análisis de la escalabilidad en los sistemas de VoD

Para lograr un sistema de VoD que sea escalable es necesario conseguir que tanto el ancho de banda servicio (número de peticiones independientes que es capaz de atender el servidor) como el ancho de banda del sistema de comunicaciones pueda crecer a medida que el sistema se amplia.

De los principales componentes de un sistema de VoD (servidor y red de transmisión) el ancho de banda del servidor siempre es mayor que el ancho de banda de red, debido a que la tecnología interna de bus ofrece una mejor relación coste / prestaciones y una mayor escalabilidad. Este crecimiento se puede conseguir mediante la inclusión de nuevos discos en configuración de RAID, con la utilización de la metodología de cluster ó la utilización de servidores paralelos (varios servidores independientes conectados a la misma red de servicio). Aún así la creación de estos servidores que puedan gestionar decenas de miles de peticiones simultáneas es compleja y muy costosa debido a las altas prestaciones requeridas.

Por otro lado, el ancho de banda de la red es menor (debido al coste asociado) convirtiéndose en el verdadero cuello de botella del sistema a la hora de su crecimiento, restringiendo considerablemente su tamaño.

La razón de la pobre escalabilidad del sistema de comunicaciones estriba en que su ancho de banda esta limitado por la tecnología disponible en cada momento. La capacidad de la red, también esta limitada por la infraestructura utilizada y sus únicas posibilidades de crecimiento pasan por la modificación de toda la infraestructura ó la inclusión de nuevas redes conectadas a un servidor central (sistemas basados en servidores-proxy) ó servidores independientes. La primera alternativa implica un alto coste debido a la necesidad de cambiar toda la infraestructura de transmisión y no garantiza que al cabo de poco tiempo el sistema se vuelva a saturar y se tenga que volver a ampliar de nuevo. La segunda alternativa implica la utilización de arquitecturas que no soportan todas las características exigidas para un sistema LVoD escalable, como ya hemos expuesto en el capítulo anterior.

En definitiva, el tamaño y crecimiento final de los sistemas de VoD dependen, en ultima instancia, de la capacidad (ancho de banda) de la red de conexión con los clientes finales. Para

poder lograr un sistema de transmisión escalable es imperativo prescindir de las limitaciones (físicas) de los sistemas centralizados (una única red ó un único servidor), ya que éstos siempre estarán limitados por la tecnología disponible en cada momento. Por lo tanto, vamos orientar el diseño hacia los sistemas distribuidos basados en servidores-proxy.

2.2.2 Topología en árbol

Nuestra primera propuesta para alcanzar un sistema totalmente escalable, consiste en expandir la técnica de servidores-proxy, hasta ahora restringida a un único nivel, mediante la utilización de una topología en árbol. Esta arquitectura la hemos denominado Proxy-Tree ó P-Tree [Cor01].

La razón de fundamentar el diseño de la arquitectura propuesta en los servidores-proxy estriba en que estas arquitecturas cumplirían la mayoría de los requisitos de los sistema LVoD, sino fuese por la utilización de componentes centralizados para la gestión del sistema y para mantener una copia de todos los contenidos del sistema. Por lo tanto, uno de nuestros objetivos consistirá en eliminar la dependencia de estos componentes centralizados, logrando un sistema verdaderamente distribuido.

Esta topología aporta al sistema una capacidad de escalabilidad ilimitada, así como una gran flexibilidad a la hora de escoger el tamaño del sistema ó su forma de crecimiento. Esta flexibilidad nos va a permitir una mayor maniobrabilidad a la hora de desplegar el sistema y en su posterior crecimiento. Sin embargo, no todos son ventajas en la topología en árbol. Hay que tener especial cuidado con el nodo inicial del árbol, ya que puede convertirse en el cuello de botella del sistema.

La topología P-Tree, mostrada en la figura 2-1, consta de una serie de niveles (4 en este caso), en función del número de redes locales que componen el sistema de VoD y del orden del árbol (binario, terciario, etc.).

Cada nivel de la jerarquía estará compuesto por una serie de redes, denominadas redes locales, que se conectan con el nivel superior del árbol. De cada una de las redes locales de un nivel, se colgarán las redes del siguiente nivel, y así sucesivamente hasta completar el último nivel. En cada red de la topología (salvo la red principal) se conectará un servidor-proxy (ó también denominado servidor local) y un subconjunto de los usuarios del sistema.

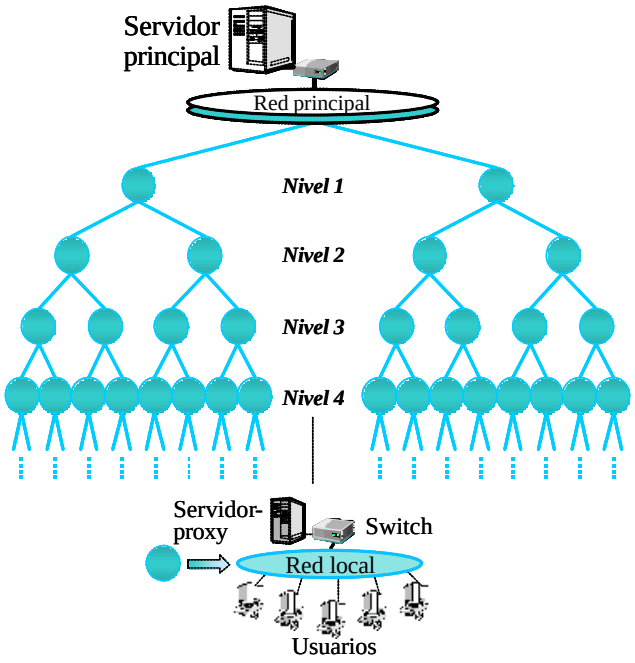


Figura 2-1. Arquitectura Proxy-Tree para un sistema LVoD.

La replicación de todos los contenidos del catálogo del sistema en cada uno de los servidores locales puede resultar económicamente inviable si el catálogo es demasiado grande. Por lo tanto, a diferencia de los sistemas de servidores independientes, nuestra arquitectura asumirá que los servidores locales dispondrán de una capacidad de almacenamiento limitada para los contenidos multimedia.

Cada uno de los servidores-proxy de forma individual no puede contener una copia completa de los videos del catálogo del sistema. Para suplir esta carencia, la arquitectura dispondrá de un servidor principal, ubicado en el primer nivel de la topología. Este servidor dispondrá de una copia completa del catálogo de contenidos multimedia.

Esta arquitectura es fácilmente escalable ya que el añadir una nueva red local únicamente implica la inclusión de un switch para separar el tráfico de los dos segmentos de red, sin tener que modificar ningún otro componente del sistema.

La topología en árbol utilizada, nos ofrece una gran versatilidad, debido a que el sistema inicialmente puede estar compuesto por cualquier número de redes y puede crecer tanto a lo largo, añadiendo nuevos niveles ó bien a lo ancho, incrementando el orden de algunos de los nodos del árbol. Hay que subrayar que la arquitectura Proxy-Tree no requiere ni que los niveles estén completos (sin embargo, es recomendable acabar de llenar un nivel antes de empezar el

siguiente, para lograr un sistema equilibrado) ni tampoco requiere que todos los niveles del árbol tengan el mismo número de redes conectadas (mismo orden).

2.2.3 Políticas de gestión de los contenidos en los servidores-proxy

En un primer análisis, hemos podido constatar que una topología jerárquica en árbol tiene cualidades, desde el punto de vista de la escalabilidad y versatilidad, ideales para una arquitectura LVoD. No obstante, independientemente del tipo de topología utilizada, se debe demostrar que la arquitectura propuesta escala sin saturar el sistema (en concreto, la red de transmisión).

Dentro de la arquitectura P-Tree, todos los servidores-proxy cuyas redes locales se encuentren situadas a una misma distancia del nodo en donde se produjo la petición definen un nivel dentro de un sistema jerárquico de cache. El tamaño de cada nivel de cache vendrá definido como la suma de las capacidades individuales de cada uno de los servidores-proxy situados dentro de una determinada distancia. Se debe subrayar que a medida que se pasa de un nivel jerárquico de cache al superior se aumenta el número de servidores-proxy a los cuales se puede acceder, pero también se incrementa el ancho de banda de red requerido para servir la petición (la petición precisará ancho de banda en cada una de las redes locales que se deban cruzar).

En la figura 2-2 se muestran los tres niveles jerárquicos de cache definidos alrededor de un nodo de la topología, podemos ver que el primer nivel esta compuesto exclusivamente por el servidor local, el segundo nivel está compuesto por 3 servidores-proxy adicionales (situados a distancia 1 del servidor-proxy en donde se produjo la petición original) y el tercer nivel está compuesto por 5 servidores-proxy (situados a distancia 2) más el servidor principal.

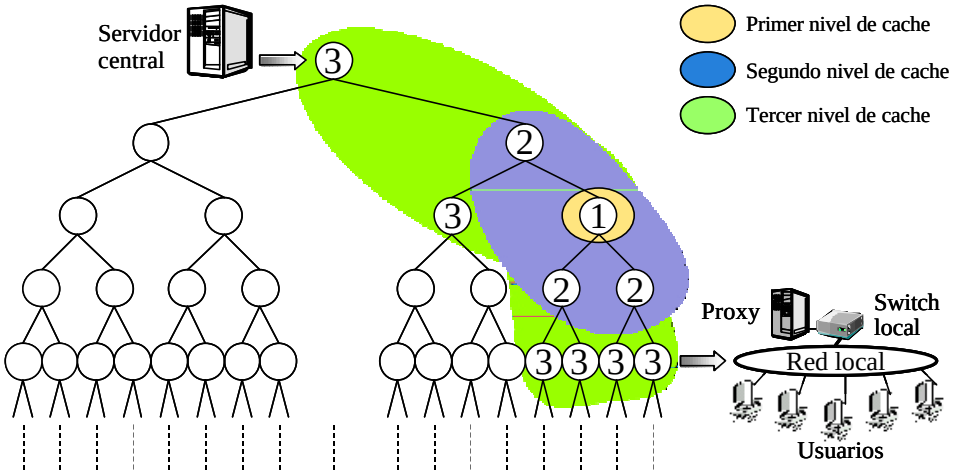


Figura 2-2. Sistema jerárquico de caching

Este sistema jerárquico propuesto tiene la ventaja de que a medida que una petición no puede ser atendida localmente por su servidor-proxy (primer nivel del almacenamiento jerárquico), ésta se puede dirigir hacia los servidores-proxy vecinos (segundo nivel del almacenamiento jerárquico) que pueden intentar atender la petición. Si ninguno de los servidores vecinos pueden gestionar la petición, entonces ésta se envía hacia los siguientes servidores (tercer nivel del almacenamiento jerárquico), y así sucesivamente hasta que se alcance un servidor-proxy puede atender la petición ó hasta que se alcance el servidor principal.

No obstante, este planteamiento tiene un grave problema. Cuando el almacenamiento de los servidores-proxy se gestiona como una cache (como se hace en el esquema tradicional de servidores-proxy), el sistema jerárquico resultante no logra mejorar la escalabilidad ni la eficiencia del sistema LVoD, tal y como se expone a continuación.

Los servidores-proxy tradicionales, gestionan su almacenamiento como una cache de contenidos multimedia, almacenando en cada momento los videos más populares. Este sistema de gestión de contenidos, implica una alta probabilidad de que todos los servidores-proxy tengan replicados los mismos videos. Por lo tanto, si una petición no puede ser servida desde su propio servidor-proxy local, entonces muy probablemente tampoco podrá ser servida por ninguno de los servidores-proxy del sistema, y se tendrá que acabar recurriendo al servidor principal.

Este comportamiento, queda reflejado en la figura 2-3, en la cual una petición que no ha podido ser atendida por el servidor de un nivel intermedio de la topología, no puede ser atendida por ninguno de los servidores-proxy y finalmente tiene que acudir al servidor principal. En este caso, el fallo requiere que el stream de servicio entre el servidor principal y el usuario deba cruzar

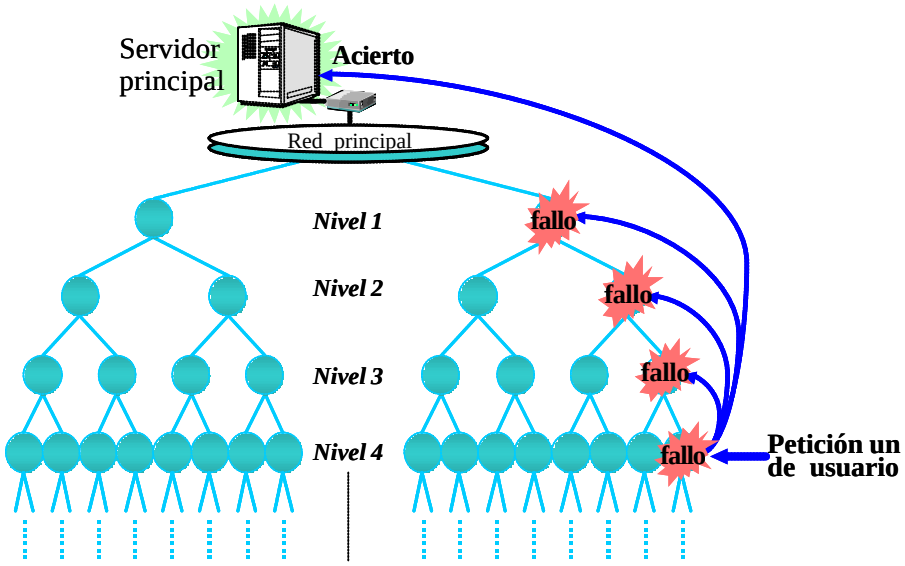


Figura 2-3. Gestión de una petición remota en un sistema jerárquico de caching

5 redes de la arquitectura (red principal + 4 redes locales). Por lo tanto, la petición requerirá 5 veces más ancho de banda que si se hubiese atendido localmente. Además, que el servidor principal sea el único responsable de atender todas las peticiones que fallan en los servidores-proxy, provoca una rápida saturación de sus recursos, limitando considerablemente la capacidad de servicio y la escalabilidad de la arquitectura.

Este mismo problema aparece en los sistemas basados en servidores-proxy de un nivel, pero no es tan grave debido a que la existencia de una topología de sólo dos niveles limita la penalización de los fallos en los servidores-proxy a únicamente el doble de ancho de banda de red de una petición local (red principal + red local).

Nuestra arquitectura requiere de una nueva organización para los servidores-proxy, que nos permita incrementar la probabilidad de acierto a medida que la petición se mueve por los distintos niveles del árbol. A continuación analizamos dos posibles alternativas para aumentar la capacidad de los servidores-proxy a la hora de gestionar las peticiones remotas: el caching cooperativo y el mirroring.

Caching cooperativo

Mediante esta técnica, el almacenamiento dedicado a la cache de una serie de servidores-proxy adyacentes se gestiona conjuntamente como si se tratase de una única cache distribuida [Wan99] [Hof99] [Gri00]. Las políticas de gestión de esta cache se realiza teniendo en cuenta el estado y los contenidos de todos los servidores-proxy que la componen. Por ejemplo, las políticas de reemplazo en la cache cooperativa tendrán en cuenta las estadísticas de acceso en todos los servidores-proxy cooperantes.

Esta técnica permite incrementar considerablemente el tamaño de la cache, pero a costa de reducir la probabilidad de acierto local. Bajo el esquema de cache cooperativo, un único servidor-proxy no dispone de los contenidos más populares, provocando un aumento del porcentaje de peticiones que se tienen que servir remotamente desde los otros servidores que componen la cache ó desde el servidor principal.

Por otro lado, ésta técnica tampoco termina de subsanar uno de los problemas asociados con las caches: una parte de los contenidos (los menos populares) nunca tendrán la posibilidad de ubicarse en alguna de las caches y se tendrán que servir siempre desde el servidor central. Un caso extremo de cache cooperativa, que permite soslayar este problema, es aquella que tiene suficiente capacidad entre todos los servidores-proxy involucrados como para almacenar todos los contenidos del catálogo. En este caso, ya no se puede hablar de una cache cooperativa (ya que la frecuencia de acceso de los contenidos no juega ningún papel en la selección de los contenidos almacenados) si no de un mirror distribuido.

En resumen, la cache cooperativa es una técnica que da buenos resultados a la hora de reducir la latencia de acceso, pero que sin embargo penaliza considerablemente el ancho de banda de red requerido para las peticiones que fallan en la cache, manteniendo la dependencia con un catálogo de contenidos centralizado. Por lo tanto, no es una buena aproximación para aportarnos la escalabilidad deseada a nuestra arquitectura.

Mirroring

La técnica de mirroring consiste en replicar los contenidos, sin tener en consideración su frecuencia de acceso (popularidad) [Sch95][Chu00]. Esta técnica es útil cuando se quiere aportar tolerancia a fallos ó bien reducir la latencia de acceso a información remota, manteniendo una copia de los contenidos más cerca del usuario final.

En nuestro caso, el mirroring también nos interesa para reducir los requisitos de ancho de banda de las peticiones que se tienen que atender remotamente y poder reducir así la saturación del servidor principal.

En los esquemas más comunes de mirroring, éstos suelen contener una copia completa de la información del sistema [Sch95]. Un ejemplo de este esquema de gestión en VoD es la arquitectura de sistemas independientes, en la cual cada servidor mantiene un mirror de los contenidos del sistema. En [Chu00] se analizan las diferencias en el rendimiento entre las políticas de caching y replicación (mirroring) de contenidos web, en función de la calidad de servicio, requerimientos de almacenamiento y perfil del tráfico.

No deseamos la utilización de un mirror completo en nuestra arquitectura, debido al alto coste en almacenamiento que implica el replicar todos los contenidos en todos los servidores-proxy. En cambio, nosotros proponemos la utilización de un esquema de mirroring distribuido, en el cual cada servidor local mantendría una proporción de los contenidos del catálogo del sistema. De esta forma, el mirror estaría distribuido entre los mirrors parciales pertenecientes a un conjunto de servidores-proxy cercanos entre si.

Mediante este esquema, el sistema dispondrá de una serie de mirrors distribuidos, cada uno de ellos formados por N/V_s servidores, siendo N el número de videos del catálogo del sistema y V_S el número de videos que caben en el mirror parcial de cada servidor proxy.

En la Figura 2-4 mostramos los distintos niveles de mirroring a los cuales se puede acceder desde cada nodo de la arquitectura P-Tree. Los contenidos almacenados en los mirrors de los servidores situados a distancia 2 de los usuarios de un determinado nodo de la arquitectura constituyen el mirror distribuido de distancia-2 de ese nodo. En el ejemplo, esto daría lugar a un mirror distribuido a distancia-2 formado por 4 servidores (local + 3 vecinos).

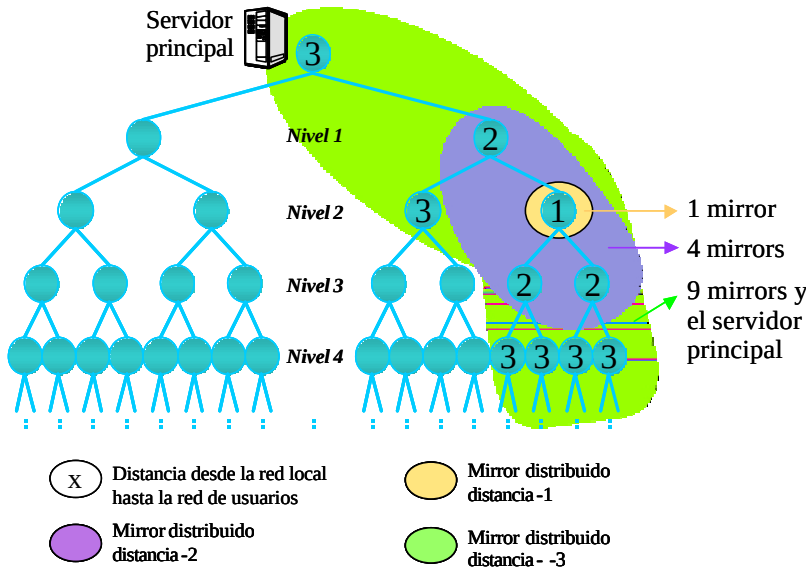


Figura 2-4. Niveles de mirroring en la arquitectura P-Tree

De forma equivalente, el mirror distribuido a distancia-3 lo componen los contenidos ubicados en los mirrors de los servidores situados a distancia-3 de cada servidor proxy. En el ejemplo este mirror estaría compuesto por 9 servidores locales más el servidor principal del sistema.

Tal y como hemos comentado, utilizando únicamente una política de caching, los contenidos menos populares tienen que ser gestionados en exclusiva por el servidor principal. Este volumen de carga dirigido a un único componente puede provocar a la larga su saturación, limitando la escalabilidad del sistema de VoD. Sin embargo, con la utilización del mirroring distribuido, la mayor parte (si no toda) de esta carga es gestionada por lo propios servidores-proxy mediante los mirrors parciales.

Por otro lado, a medida que se incrementa el número de redes locales que componen el sistema, también incrementamos el número y tamaño de los mirrors distribuidos (gracias al aumento de la capacidad de almacenamiento total disponible en el sistema LVoD). De esta forma, este aumento en la capacidad de almacenamiento permite compensar la carga de trabajo adicional asociada con los nuevos usuarios incorporados, logrando un sistema más escalable.

La utilización de mirroring distribuido no es suficiente por si mismo para resolver todos los problemas asociados con un sistema LVoD escalable. Al igual que ocurre con el caching cooperativo, la utilización de un mirror distribuido reduce el porcentaje de peticiones que se pueden servir localmente e incrementa el número de peticiones que se tienen que atender desde servidores-proxy remotos. Las peticiones remotas pueden a llegar a necesitar el doble ó el triple del ancho de banda de red (consumen recursos de red en cada red de la arquitectura que tienen

que cruzar) de una petición local. Por lo tanto, una excesiva utilización de éstas, puede reducir considerablemente el rendimiento. Para solucionar estos inconvenientes, proponemos la utilización conjunta de las dos técnicas: el caching y el mirroring.

Caching + Mirroring

Nuestra propuesta final para la gestión de los contenidos en la arquitectura P-Tree, consiste en dividir el espacio de almacenamiento disponible en cada uno de los servidores-proxy en dos partes, ver figura 2-5. Una porción del espacio de disco seguirá funcionando como una cache, almacenando los videos más populares. El resto del espacio se utilizará para hacer un mirror distribuido de los contenidos que conforman el catálogo del sistema.

Para facilitar el análisis de la arquitectura propuesta, la gestión de los contenidos en los servidores-proxy se realizará trabajando siempre con los objetos multimedia (videos) completos. Esta simplificación no supone que en la arquitecturas propuestas no soporten políticas de gestión fragmentada de los contenidos, como por ejemplo: *prefix-caching* [Sen99b] ó *segmented-caching* [Kun01]. Consideramos que la utilización de una ú otra política de caching no afecta a los principales parámetros de rendimiento de las arquitecturas propuestas y hemos optados por demorar el análisis de estas políticas para líneas de investigación futuras.

Cada uno de los esquemas de gestión de contenidos en los servidores-proxy tiene un papel bien diferenciado. El esquema de caching, se utiliza primordialmente para incrementar el número de peticiones gestionadas localmente. Debido a las características especiales de los contenidos multimedia, unos pocos videos pueden recibir un gran porcentaje de las peticiones de los usuarios, lo cual permite que la política de caching alcance una alta eficiencia con muy pocos recursos. El principal defecto de este esquema de gestión es la replicación de contenidos inherente a la política y su escasa efectividad para la gestión de las peticiones remotas.

El esquema de mirroring esta orientado a reducir la distancia requerida para atender las peticiones remotas, reduciendo el ancho de banda de red consumido por éstas.

Con ambos esquemas, trabajando en colaboración conseguimos alcanzar los siguientes objetivos:

Figura 2-5. Esquemas de gestión de contenidos en los servidores-proxy

1. Reducir la carga del servidor principal.

   Al utilizar el esquema de cache, incrementamos el número de peticiones servidas localmente, por lo tanto se reduce el volumen de carga que debe soportar el resto de servidores del sistema.

   Además, gracias al mirror distribuido, todas aquellas peticiones generadas en nodos situados más allá de una determinada distancia del servidor serán atendidas por los mirrors y no alcanzarán al servidor principal, reduciéndose así su saturación.

2. Reducir la distancia media de servicio.

   El mirroring distribuido permite reducir la distancia de servicio de aquellas peticiones que han fallado en su propio proxy. A medida que se incrementa la distancia de servicio también se incrementa el tamaño del mirror distribuido y, por tanto, la probabilidad de acierto también es mayor.

3. Mayor flexibilidad del sistema LVoD.

   La combinación de dos esquemas de gestión tan complementarios nos permite adaptar la distribución del almacenamiento entre ambos en función de los recursos disponibles y las necesidades del sistema.

4. Incrementar la tolerancia a fallos.

   El fallo de uno de los nodos del árbol (inclusive el servidor principal) no impide que el resto del sistema pueda seguir funcionando gracias a la utilización de los mirrors distribuidos.

2.2.4 Funcionalidad de la arquitectura Proxy-Tree

Los servidores-proxy utilizados en la literatura, tienen un comportamiento similar a los servidores-proxy de Internet, los cuales monitorizan las peticiones de los usuarios locales, capturando y sirviendo directamente aquellas peticiones que hacen referencia a contenidos almacenados en su cache.

Esta funcionalidad, en principio, es coherente con la arquitectura propuesta, en la cual la gestión del sistema esté centralizada en el servidor principal. Ahora bien, utilizando una gestión centralizada, el servidor principal toma todas las decisiones sobre cómo y desde qué servidor se atenderán las peticiones que no pueden ser gestionadas localmente.

La gestión centralizada permite que la implementación de los servidores-proxy sea más sencilla, pero implica una mayor saturación del servidor principal y una menor tolerancia a fallos (la caída del servidor ó de la red principal puede provocar la parada de todo el sistema).

Un sistema de gestión centralizado choca con nuestra intención de lograr un sistema tolerante a fallos y sin cuellos de botella, por lo que vamos a optar por una gestión distribuida de la arquitectura P-Tree.

La integración de los servidores-proxy en una arquitectura totalmente distribuida implica dotar a los servidores-proxy de una nueva funcionalidad referida a la gestión distribuida del sistema. Esta nueva funcionalidad incrementa la complejidad de los servidores-proxy hasta el punto de poder confundirlos con un servidor estándar de VoD.

La principal diferencia entre los servidores-proxy de la arquitectura P-Tree y los servidores VoD convencionales estriba en la capacidad de almacenamiento limitada (suficiente para dar cabida a un subconjunto de los videos del catálogo del sistema) y la utilización de una política de gestión de los contenidos basados en el caching de los videos más populares. *A lo largo de la memoria vamos a utilizar indistintamente la nomenclatura de servidores-proxy ó servidores locales para referirnos a este híbrido entre un servidor y un proxy.*

El primer nodo de la topología se puede comportar como un nodo más de la arquitectura ó bien realizar la misma funcionalidad que el nodo principal (servidor + red principal) en los sistemas de servidores-proxy de un nivel. En este último caso, la única diferencia del servidor principal respecto al resto se servidores-proxy estriba en que el servidor principal es el único que mantiene una copia completa del catálogo de contenidos. En el caso de que el primero nodo de la topología tenga la misma funcionalidad que cualquier otro nodo del sistema, entonces nos encontramos ante un sistema homogéneo en el cual todos los nodos tienen los mismos componentes y realizan la misma funcionalidad.

En la figura 2-6 mostramos el diagrama básico de gestión de los servidores en la arquitectura P-Tree. Como se puede apreciar, al servir una petición en la arquitectura se pueden dar dos casos diferentes: que la petición se pueda gestionar localmente por parte del proxy asociado con el cliente ó que la petición se deba atender desde un proxy diferente.

Cuando un proxy ó servidor local recibe una petición de un usuario debe comprobar si el contenido solicitado esta disponible en su almacenamiento y si tiene suficiente recursos (ancho de banda de servicio y de red) para asegurar el servicio de la petición con la calidad de servicio requerida. En el caso que todos los requisitos se cumplan, la petición se atiende por el servidor local a través de la red local y la red del cliente (petición local).

Si por cualquier razón (el servidor local no dispone del contenido pedido, ó no hay suficiente ancho de banda de red ó servicio) la petición no se puede atender localmente y es necesario redirigirla hacia otro servidor que si pueda servirla (petición remota).

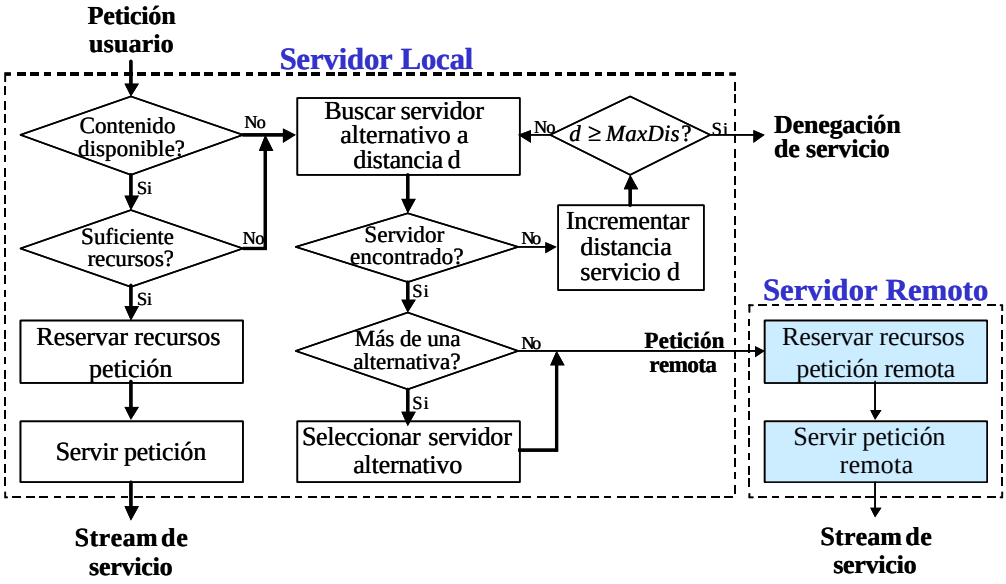


Figura 2-6. Diagrama de la gestión de los servidores-proxy en la arquitectura P-Tree

La selección del servidor alternativo se realizará teniendo en cuenta la distancia del servidor remoto. Primero se comprobará si hay algún candidato entre los servidores adyacentes (a distancia-1 desde el servidor origen), si no es el caso, se analizan los servidores situados a distancia-2 y así sucesivamente hasta que se localiza algún servidor que pueda atender la petición. Al dar más prioridad a los servidores cercanos, se busca reducir el ancho de banda de red consumido por las peticiones remotas ya que los recursos de red consumidos por este tipo de peticiones puede llegar a ser el doble ó incluso el triple (dependiendo de la distancia entre los dos servidores involucrados) del requerido por una petición local.

La búsqueda de un servidor alternativo requiere disponer de información sobre el estado (contenidos y recursos disponibles) de los servidores adyacentes. Este estado consistirá en información sobre los contenidos almacenados en el servidor remoto, ancho de banda de servicio disponible y ancho de banda disponible en la red local del servidor remoto. Esta información se puede obtener mediante un proceso de encuesta ó bien manteniendo de forma local el estado de cada uno de los servidores adyacentes. Seguidamente se explicará el funcionamiento de ambos métodos.

Mediante el método de encuesta, cada vez que se necesita redirigir una petición, se envía un mensaje de control a cada uno de los servidores situados a una misma distancia del servidor origen. El mensaje de control se utilizará para preguntar a los distintos servidores remotos si tienen suficientes recursos para atender la petición. Los servidores remotos contestan al servidor

origen afirmativamente ó negativamente, adjuntando la información precisa para que el servidor local pueda tomar una decisión.

En el caso que todos los servidores remotos de una determinada distancia respondan negativamente, se repite el proceso de encuesta, esta vez con los servidores situados a la siguiente distancia. Si uno ó más servidores responden afirmativamente, el servidor local escoge uno de ellos atendiendo a diferentes criterios (balanceo de carga / trafico, compartición de recursos, etcétera).

El método de encuesta puede generar un considerable aumento de los mensajes de control entre los servidores de la arquitectura. Otra alternativa, consiste en disponer de una tabla con el estado de los recursos de todos los servidores del sistema. Esta tabla se actualizará mediante información adjunta a los mensajes de control intercambiados entre los componentes de la arquitectura ó bien bajo demanda en el caso que la información disponible esté obsoleta.

Una vez localizado un servidor adecuado, se le redirige la petición del usuario. A partir de este instante el servicio de la petición se realiza directamente entre el servidor remoto y el cliente, sin que tenga que intervenir el servidor local del usuario. Únicamente, en el supuesto que al servidor local le interese guardar en su cache el contenido servido al usuario, podrá éste intervenir en la trasmisión. En este caso se utilizará una técnica de multicast para servir con el mismo stream al usuario y al servidor local. En el caso de que no haya ningún servidor adecuado se deniega la petición al usuario.

2.3 Modelo analítico de las arquitecturas basadas en servidores-proxy

Para evaluar la escalabilidad de una arquitectura de VoD es necesario estudiar como evoluciona el rendimiento del sistema, entendido como la capacidad de servicio del mismo, a medida que crece mediante la inclusión de nuevos nodos de servicio y usuarios. La realización de este análisis implica un cálculo de la capacidad de servicio del sistema y de los requisitos de cada uno de los componentes (redes / servidores) de las arquitecturas estudiadas: un nivel de servidores-proxy y P-Tree.

Para el cálculo de los distintos parámetros de rendimiento, nos vamos a basar en la definición de un modelo analítico que nos permitirá evaluar la capacidad de servicio y la escalabilidad de la arquitectura P-Tree, y comparar los resultados obtenidos con otras aproximaciones LVoD.

El estudio analítico de la arquitectura P-Tree lo vamos a realizar de forma progresiva. Inicialmente definiremos un modelo para evaluar analíticamente la escalabilidad y el rendimiento de un sistema de servidores-proxy de un nivel. Posteriormente adaptaremos este modelo para que tenga en cuenta un sistema jerárquico de servidores-proxy (Proxy-Tree utilizando únicamente caching) y finalmente introduciremos el uso combinado de caching y mirroring dentro de los servidores de la arquitectura.

La evaluación analítica los sistemas mencionados (basados en servidores-proxy) requiere la definición de un modelo analítico con las siguientes características: Se supone la utilización de redes no segmentadas en las distintas arquitecturas, cada servidor-proxy y red local dispondrá de un ancho de banda de B_c Mb/s, tamaño del proxy suficiente para almacenar V_s películas de los M videos disponibles en el sistema, ancho de banda del servidor principal y de la red principal de B_p Mb/s.

En la realización de este análisis vamos suponer la utilización una política de unicast, de forma que cada usuario tiene asignado su propio stream de trasmisión para los contenidos solicitados. Este supuesto es valido ya que el estudio de la escalabilidad y la capacidad de servicio del sistema (entendida como el número de streams independientes que puede gestionar) es independiente de la utilización ó no de políticas de compartición de recursos. Los resultados obtenidos mediante el modelo analítico serán independientes de que después se pueda utilizar políticas de compartición de streams (broadcast, multicast, etc.) para incrementar la eficiencia y el número de clientes finales del sistema.

Para poder medir la eficiencia de las distintas arquitecturas LVoD, vamos a calcular el *ancho de banda efectivo del sistema*, definido como el número de streams simultáneos que se pueden servir utilizando los mismos recursos (ancho de banda de red y de servicio). El ancho de banda efectivo dividido entre el ancho de banda requerido para servir una petición, nos indicará el número de streams que es capaz de soporta la arquitectura.

Para estimar la capacidad de crecimiento del sistema, evaluaremos el volumen y la distribución del tráfico generado en el sistema. Esta medida nos aportará una idea de la limitación del sistema con respecto al número de usuarios que puede admitir y de su grado de escalabilidad.

Sistema de servidores-proxy de un nivel

Como ya hemos comentado las arquitecturas de servidores-proxy de un nivel, se componen de una serie de redes locales conectadas todas directamente al servidor principal mediante su red.

De forma genérica, el ancho de banda efectivo (B_e) de un sistema basado en servidores-proxy se evaluará como el ancho de banda máximo disponible en el sistema (B_m) menos el ancho de banda adicional (B_{fp}) requerido por los fallos de los servidores-proxy. Los fallos en los servidores-proxy tienen que ser gestionados por el servidor principal, lo cual implica utilizar la red principal tanto como la red local. Dichas peticiones remotas requieren el doble de ancho de banda que una petición local y por lo tanto implican una reducción del ancho de banda efectivo del sistema. De este modo:

$$B_e = B_m - B_{fp} \qquad (2.1)$$

Este ancho de banda máximo disponible en el sistema se obtendrá como la suma del ancho de banda de todas las redes que forman el sistema LVoD, según la siguiente expresión:

$$B_m = B_p + B_c \cdot n \qquad (2.2)$$

siendo B_p el ancho de banda de la red principal, B_c el ancho de banda de las redes locales y n el número de redes locales.

Para obtener el ancho de banda adicional (B_{fp}) requerido por los fallos de los servidores-proxy necesitamos calcular la probabilidad de fallo de los mismos. Debido a que los servidores-proxy en esta arquitectura gestionan los contenidos exclusivamente mediante el esquema de caching, asumiremos que los servidores-proxy almacenarán en su cache los V_c videos completos más populares.

Para poder definir cuales son los videos más populares del sistema VoD, es necesario modelar la frecuencia de acceso (ó la probabilidad de selección) de cada uno de los videos del catálogo del sistema. Dan y Sitaran analizaron la frecuencia de acceso a los videos de un video-club [Dan94] [Dan96] y concluyeron que la popularidad de los contenidos de un servidor VoD se pueden modelar mediante una distribución *Zipf* [Zip49] con un grado de popularidad (*skew factor*) de z. A partir de la distribución *Zipf* y del número de contenidos del catálogo (M) se puede calcular la probabilidad de acceso de cada uno de los contenidos (f_i) mediante la siguiente expresión:

$$f_i = \frac{1}{i^z \cdot \sum_{j=1}^{M} 1/j^z} \qquad (2.3)$$

A partir de la expresión (2.3) la figura 2-7 muestra la distribución de las probabilidades de acceso para cada uno de los contenidos multimedia en un sistema con 100 videos y un grado de popularidad del 0.729. Se puede constatar que los contenidos están ordenados según su

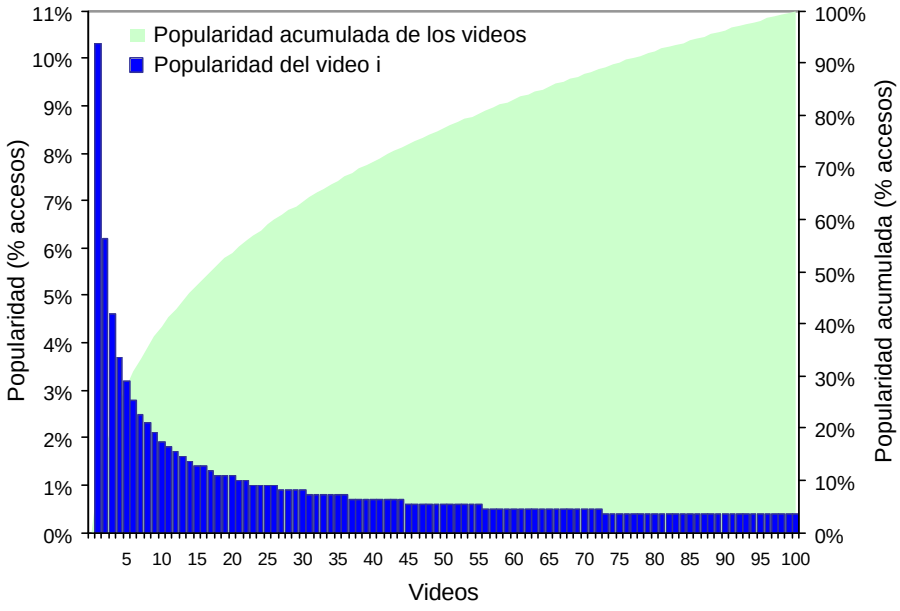


Figura 2-7. Distribución Zipf de popularidad en los videos

popularidad: los videos con un índice menor tienen una mayor frecuencia de acceso. También podemos observar que un reducido grupo de contenidos reciben la mayoría de los accesos. Por ejemplo, los 20 primeros videos reciben más del 53% de las peticiones de los usuarios.

Mediante la expresión (2.3), podemos calcular la probabilidad de que una petición pueda ser atendida por el servidor-proxy (acierto en el proxy) como la suma de las probabilidades individuales de los V_c contenidos más populares, y mediante su inverso podemos evaluar la probabilidad de fallo al acceder a un servidor-proxy (p_{fp}) según la siguiente formula:

$$p_{fp} = 1 - \sum_{i=1}^{Vc} f_i \qquad (2.4)$$

De esta forma podemos calcular el ancho de banda adicional requerido en el sistema debido a los fallos de los servidores-proxy como la probabilidad de fallo en cada proxy (p_{fp}) multiplicado por el tráfico generado en todas las redes locales del sistema ($B_c \cdot n$), es decir:

$$B_{fp} = n \cdot B_c \cdot p_{fp} \qquad (2.5)$$

Esta expresión mide el trafico enviado por los servidores-proxy hacia el servidor y la red principal, y por lo tanto, identifica el ancho de banda mínimo requerido en la red principal para que el sistema no se sature con los fallos en los servidores proxy.

Sustituyendo las expresiones (2.2) y (2.5) en la expresión (2.1), el ancho de banda efectivo de una arquitectura de servidores-proxy de un nivel es:

$$B_e = B_p + n \cdot B_c - n \cdot B_c \cdot p_{fp} = B_p + n \cdot B_c \cdot \left(1 - p_{fp}\right) \qquad (2.6)$$

Sistema jerárquico de servidores-proxy

Habiendo analizado la efectividad de los sistemas basados en servidores-proxy de un nivel, a continuación nos planteamos como podemos extender este análisis para contemplar los sistemas basados en una topología jerárquica de servidores-proxy (arquitectura P-Tree utilizando exclusivamente caching en los servidores).

Para facilitar este estudio asumiremos una topología en árbol completa (todos los niveles están llenos) con *L* niveles en los cuales el orden (*o*) identifica el número de redes locales conectadas en cada uno de los nodos del árbol.

Del modelo anterior, el único parámetro que se ve afectado con la modificación de la topología, es el ancho de banda adicional de red requerido debido a los fallos en los servidores-proxy de las redes locales (expresión 2.5). En esta topología, a medida que una petición no puede ser servida por el proxy de un nivel, se accederá al siguiente y así sucesivamente. A medida que se accede al siguiente nivel de cache se incrementa el ancho de banda requerido para servir la petición. Este coste estará en función del número de niveles (distancia) que tiene que cruzar la petición desde el cliente hasta el servidor ó el proxy que la atienda.

En una primera aproximación, para evaluar este coste (B_{fp}) podríamos asumir que el coste de los fallos de las caches se pueden obtener sumando los fallos producidos en cada uno de los niveles. Asi, el coste adicional generado desde un nodo de un determinado nivel, se puede calcular como el porcentaje del tráfico de su red (B_c) que ha fallado en su proxy y que es atendido desde servidores-proxy situados a distancia 1 ($B_c \cdot p_{fp}$), más el porcentaje del trafico que no se sirve desde los servidores-proxy de distancia 1 y que se atienden en los servidores-proxy de distancia 2 ($B_c \cdot p_{fp} \cdot p_{fp}$) y así sucesivamente hasta llegar al nivel 0, en el cual se encuentra el servidor principal que será el que atenderá en última instancia la petición. El coste generado por un nodo de un nivel se tiene que multiplicar por el número de nodos de ese nivel para obtener el ancho de banda requerido en el mismo. A través de este planteamiento, el ancho de banda requerido para atender los en los servidores-proxy (B_{fp}) se define como:

$$B_{fp} = B_c \cdot \sum_{l=1}^{L} o^l \sum_{d=1}^{l} p_{fp}^d \qquad (2.7)$$

Sin embargo, esta expresión no es realista debido a que asume que la probabilidad de acierto en los servidores-proxy de nivel superior es la misma que en el proxy de su red local. Esto es totalmente falso, las caches de todos los servidores-proxy tendrán probablemente los mismos contenidos (los más populares), generando un elevado nivel de redundancia de información. La replicación de contenidos entre las caches repercute negativamente en la probabilidad de acierto de los servidores-proxy, ya que si una petición ya ha fallado en un proxy, muy probablemente volverá a fallar en los restantes. Por lo tanto es más factible que la probabilidad de acierto en los servidores-proxy situados a distancia mayor de 1 de donde se genero la petición, sea prácticamente nula.

Teniendo en cuenta la reflexión anterior, una forma más realista para calcular el coste adicional en este sistema jerárquico es asumir que las peticiones que no pueden ser atendidas por su proxy local, lo son desde el servidor principal y por tanto, habrá que considerar un coste proporcional a la distancia (l) que le separa del servidor principal. Por lo tanto, en este caso el ancho de banda requerido para los fallos vendrá definido por:

$$B_{fp} \approx B_c \cdot \sum_{l=1}^{L} o^l \cdot p_{fp} \cdot l \qquad (2.8)$$

Esta formula nos demuestra que no tiene sentido esta estructura jerárquica de servidores-proxy, ya que no aumenta la probabilidad de acierto de los proxy y sí se incrementa la distancia que tienen que recorrer las peticiones cuando se falla, con el consiguiente aumento de su penalización sobre el ancho de banda efectivo del sistema.

Utilizando la expresión (2.8), en la figura 2-8 mostramos el ancho de banda de red requerido para atender a los fallos en los servidores-proxy (gráfico 2) a medida que el sistema crece. Podemos comprobar que este ancho de banda requerido crece mucho más rápido que el ancho de banda de red total (gráfico 1) disponible en el sistema, lo cual demuestra la escasa viabilidad de esta aproximación para una arquitectura escalable de LVoD.

A continuación vamos a estudiar la viabilidad de esta topología mediante la inclusión de un mirror en los servidores-proxy con el fin de reducir la distancia de servicio de las peticiones remotas y disminuir la saturación del servidor principal.

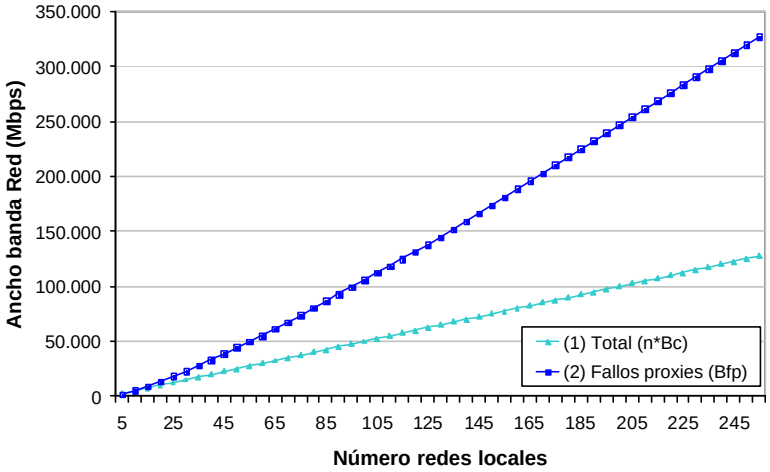


Figura 2-8. Saturación del sistema jerárquico de servidores-proxy

Arquitectura Proxy-Tree

Bajo el enfoque propuesto para la gestión de los servidores-proxy en la arquitectura P-Tree, su espacio de almacenamiento no se modifica sino que se distribuye entre dos esquemas diferentes: un porcentaje se continua gestionando como una cache para almacenar los contenidos más populares y el resto se utiliza como mirror de una porción de los videos del catálogo del sistema.

Bajo este esquema la probabilidad de acierto de las peticiones que han fallado en su propio servidor-proxy es mayor. A medida que se aumenta la distancia de servicio también se incrementa el tamaño del mirror distribuido y por lo tanto la probabilidad de que la petición remota sea atendida.

En la figura 2-9 mostramos, sobre la topología de la arquitectura P-Tree, el número de servidores-proxy alternativos a medida que se incrementa la distancia de servicio. Podemos ver como una petición que falle en su servidor local situado en el segundo nivel de la topología puede acceder a 3 servidores-proxy situados a distancia 2 (del usuario), si éstos no pueden atender la petición entonces tendrá que acceder a los 6 servidores-proxy situados a distancia 3 y así sucesivamente hasta que su petición sea atendida por un proxy ó se alcance el servidor principal.

Para facilitar el análisis asumiremos que la distribución de las películas en cada uno de los mirror se realiza de forma equitativa y de esta forma la probabilidad de acierto en los mirrors es constante e independientemente de su posición en la jerarquía y se puede aproximar dividiendo la probabilidad total (1) entre el número de contenidos del catálogo del sistema:

$$p_{am} = \frac{1}{N} \qquad (2.9)$$

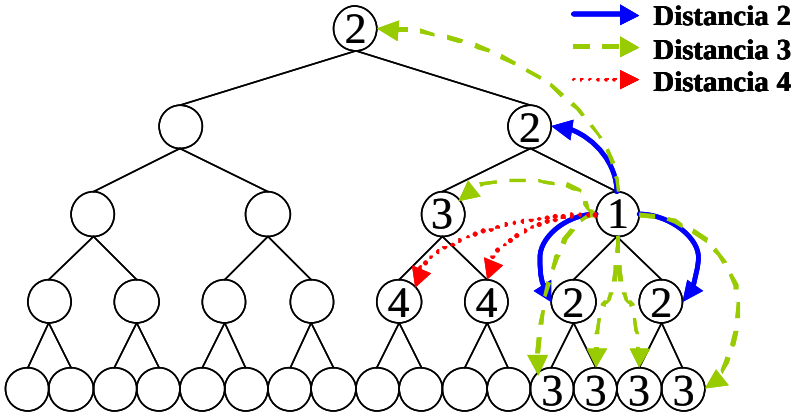


Figura 2-9. Esquema de distribución jerárquica de los fallos en los proxys

A partir de esta suposición, la probabilidad de fallo de una petición realizada en el nivel *l* y accediendo a todos los servidores-proxy situados a una distancia no superior a *d,* viene dado por la expresión:

$$p_{fm}(l,d) = 1 - \left(p_{ac} + p_{am} \cdot V_M \cdot Nd(l,d)\right) \qquad (2.\mathbf{10})$$

Siendo:

p_{ac}	Probabilidad de acierto en la cache del proxy ($1-p_{fp}$).
p_{am}	Probabilidad media de acierto en un video del mirror del proxy.
V_M	Capacidad del proxy reservada para el mirror (en videos).
$Nd(l,d)$	Número de servidores a distancia no superior a *d* del nivel *l*.

Esta expresión nos indica que la probabilidad de acierto a distancia 1 (proxy situado en la propia red local en donde se genero la petición) es la probabilidad acumulada de que la petición se pueda servir desde la cache ó desde el mirror del proxy, mientras que para distancias mayores solo se considera la probabilidad de acierto de los mirrors en los proxy situados a la misma distancia (asumimos que la efectividad de la cache para atender peticiones remotas es nula).

De esta forma, el ancho de banda necesario para los fallos en los servidores-proxy (cache+mirror) seria el siguiente:

$$B_{fp} = B_c \cdot \sum_{v=1}^{L} o^v \cdot \sum_{d=0}^{v} p_{fm}(v,d) \qquad (2.\mathbf{11})$$

sustituyendo esta expresión en la expresión (2.1) se obtiene que el ancho de banda efectivo es:

$$B_e = B_m - B_c \cdot \sum_{v=1}^{L} o^v \cdot \sum_{d=0}^{v} p_{fm}(v,d) \qquad (2.12)$$

Por otro lado, podemos evaluar el tráfico (peticiones) que recibirá el servidor principal y que caracteriza la sobrecarga en la red principal como:

$$S_p = B_c \cdot \sum_{v=0}^{L} o^v \cdot \frac{p_{fm}(v,v)}{r_s(v)} \qquad (2.13)$$

Esta expresión implica que a medida que se incrementa la distancia entre las redes donde se producen las peticiones y el servidor principal, también se incrementa el número de servidores-proxy que las pueden atender, reduciéndose así el volumen de tráfico que llega a la red principal.

El parámetro r_s identifica el porcentaje de peticiones gestionadas por el servidor principal. Si el tráfico generado por los fallos en los servidores-proxy situados a una distancia-v se distribuyen de forma equitativa entre los servidores situados a una misma distancia, entonces r_s puede ser evaluada como:

$$r_s(v) = Nd(v,v) \qquad (2.14)$$

Sin embargo las peticiones no siempre pueden ser distribuidas de forma equitativa entre los distintos servidores-proxy. Los servidores-proxy solo contienen una pequeña porción de los videos del sistema y no son capaces de gestionar tantas peticiones como el servidor principal (que dispone de todo el catálogo de contenidos). Para eliminar esta restricción, proponemos la utilización de otra distribución de los fallos, basada en la probabilidad de acierto del mirror distribuido a esa distancia, que viene dada por la expresión:

$$r_s(v) = 1 + \frac{p_{fm}(v,v) - p_{fm}(v,v+1)}{p_{fm}(v,v)} \qquad (2.15)$$

2.4 Análisis de escalabilidad

En este apartado vamos a utilizar el modelo analítico definido anteriormente para evaluar la escalabilidad de la arquitectura basada en servidores-proxy de un nivel y la arquitectura P-Tree.

Durante éste análisis se asume que el sistema homogéneo de LVoD en el cual todos los servidores y redes locales tienen el mismo ancho de banda. Las principales características son las siguientes: el ancho de banda de la red principal y de las redes locales es de 500 Mb/s, se dispone de 100 videos diferentes y un patrón de acceso modelado mediante una distribución Zipf con un factor de skew de 0.729 (grado de popularidad más frecuentemente utilizado en la literatura para modelar los sistemas de VoD [Dan96]).

2.4.1 Escalabilidad de la arquitectura de un nivel de servidores-proxy

Usando las expresiones (2.5) y (2.6), la figura 2-10 muestra los principales parámetros de comportamiento de los sistemas basados en servidores-proxy de un nivel cuando el sistema escala. Las gráficas muestran el ancho de banda efectivo del sistema (gráficas 1, 3 y 5) y el trafico recibido por la red principal (gráficas 2, 4 y 6) a medida que el sistema crece (mediante el incremento del número de servidores-proxy conectados al sistema).

Mediante este estudio queremos analizar el comportamiento del sistema, en concreto la saturación de la red, cuando crece. Para poder medir la influencia de la capacidad de almacenamiento de los servidores-proxy sobre la escalabilidad, se han obtenido resultados para distintos tamaños de cache. Así, la figura muestra los resultados utilizando servidores-proxy con capacidad para almacenar el 20%, 30% y 40% de los videos del sistema.

En los resultados mostrados podemos comprobar como a medida que aumentamos el número de servidores-proxy de la arquitectura, se aumenta el ancho de banda efectivo del sistema (gráficas 1, 3 y 5) y por ende el número de usuarios que se pueden servir.

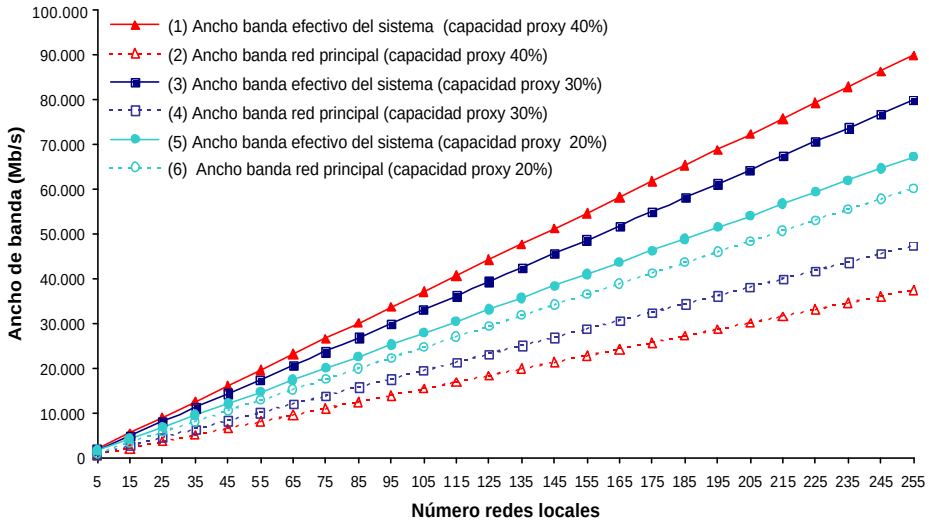


Figura 2-10. Escalabilidad en los sistemas basados en servidores-proxy de un nivel

Sin embargo, éste incremento en la capacidad del sistema se consigue a costa de incrementar los requisitos de ancho de banda de la red y del servidor principal. Así, podemos observar en las gráficas 2, 4 y 6, que el ancho requerido por la red principal se incrementa linealmente con el número de redes locales del sistema.

El aumento de los requisitos de los componentes centralizados de la arquitectura (servidor principal y red principal) ocurre independientemente de la capacidad de almacenamiento de los servidores-proxy. Tal y como podemos observar en la figura, el aumento de la capacidad de almacenamiento únicamente reduce la pendiente de crecimiento de la saturación de éstos componentes. Además, la efectividad del nuevo almacenamiento añadido es cada vez menor debido a que los nuevos contenidos incluidos en la cache tienen cada vez menor frecuencia de acceso.

Como consecuencia de estos resultados podemos concluir que *la escalabilidad de las arquitecturas de servidores-proxy de un nivel esta limitada por la capacidad de servicio del servidor principal y el ancho de banda de la red principal.*

Con respecto a la eficiencia de estas arquitecturas hacemos constar la importancia del tamaño de la cache sobre el rendimiento final del sistema. Utilizando servidores-proxy con capacidad para el 20% de las películas (gráficas 5 y 6), la mayor parte del ancho de banda efectivo obtenido (67.000 Mb/s para un sistema con 254 redes locales), proviene del ancho de banda de la red principal (60.000 Mb/s). En sistemas con servidores-proxy con una capacidad pequeña, resulta más rentable la conexión de los clientes directamente a la red principal que utilizar redes locales con servidores-proxy. Un sistema centralizado con los mismos recursos tendría un ancho de banda efectivo de 60000 Mb/s, con el ahorro del coste de todas las redes locales.

Sin embargo, cuando la capacidad de los servidores-proxy es lo suficientemente grande (tamaños del 30% y 40%), se incrementa el acierto en la cache del proxy y se reduce el acceso a la red del servidor principal. Por ejemplo, utilizando servidores-proxy con tamaño del 40% con 254 redes locales (gráficas 1 y 2) se obtienen anchos de banda efectivos de 90.000 Mb/s, requiriendo solamente una red principal con capacidad de 37.000 Mb/s.

2.4.2 Escalabilidad de la arquitectura P-Tree

Para demostrar que la arquitectura propuesta es escalable, tenemos que garantizar que la ampliación del sistema no modifica los requerimientos (ancho de banda) de los elementos existentes ó bien que el aumento de los requerimientos esta acotado. Para cumplir este objetivo tenemos que verificar que el volumen de trabajo / tráfico soportado por cualquiera de los

componentes del sistema queda acotado a medida que el sistema escala. Ahora bien, en los sistemas jerárquicos el elemento que puede recibir una mayor carga es el que se encuentra situado en el nivel superior de la jerarquía, es decir el servidor y la red principal en nuestro caso. Por lo tanto, inicialmente debemos demostrar que la red principal y el servidor principal no se saturan cuando el sistema crece.

En la figura 2-11 mostramos el ancho de banda efectivo (B_e) según la expresión (2.11) y la sobrecarga de la red principal (S_p) definida por la expresión (2.12), para sistemas basados en topologías con árboles binarios y utilizando servidores-proxy con una capacidad para el 30% y 40% de los contenidos del sistema.

Si analizamos el comportamiento de la arquitectura se puede observar que a medida que se incrementan el número de redes locales conectadas, la sobrecarga en la red principal no solo no crece, sino que inicialmente disminuye, para posteriormente estabilizarse (alrededor de los 70 Mb/s, gráfica 3, con servidores-proxy con tamaño del 40% y en 150 Mb/s, gráfica 4, para servidores-proxy con un tamaño del 30%), independientemente del número de niveles que se añadan.

Para completar el análisis de la escalabilidad, debemos garantizar que los resultados obtenidos anteriormente se pueden extrapolar directamente al resto de niveles y por lo tanto, podemos finalmente garantizar que el crecimiento del sistema no satura la arquitectura. Necesitamos verificar que los requisitos de ancho de banda de las redes locales están acotados.

La figura 2-12 muestra el tráfico soportado por las redes locales de cada uno de los niveles de la arquitectura P-Tree a medida que se aumenta el tamaño del sistema (número de redes

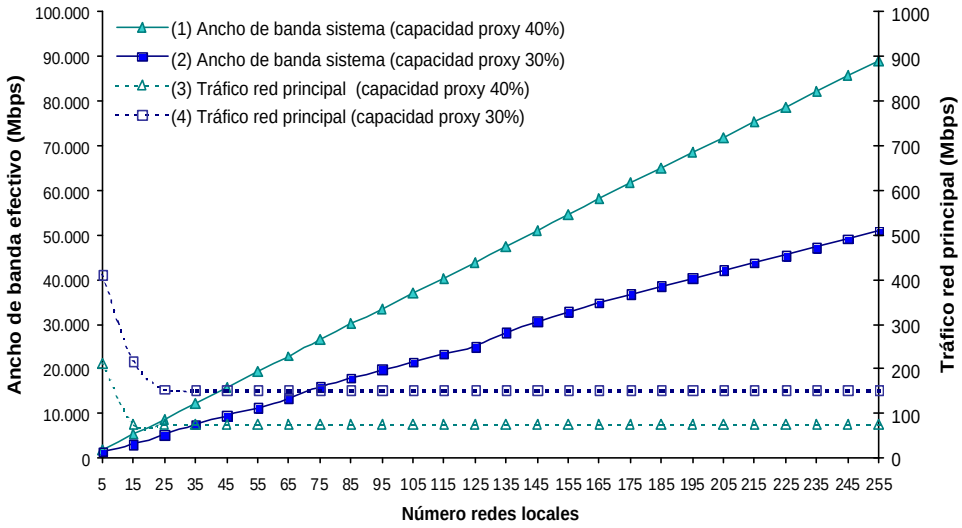


Figura 2-11. Escalabilidad de la red principal en la arquitectura P-Tree

locales). Si nos fijamos en el tráfico soportado por las redes del tercer nivel de la topología, podemos comprobar que pasa por diversas fases.

Inicialmente el tráfico de las redes del tercer nivel se estabiliza alrededor de los 660 Mb/s. Esta etapa dura justo hasta que se completan todas las redes de este nivel. En la siguiente fase, se comienza a incorporar redes locales al siguiente nivel (redes 15 a 31), lo cual provoca un incremento considerable del tráfico de las redes del tercer nivel, hasta alcanzar un máximo cercano a los 1.100 Mb/s. Este efecto se debe a que las nuevas redes incorporadas en el nivel inferior (cuarto nivel) no tienen más remedio (al no tener hijos) que enviar todo el tráfico asociado con los fallos en los servidores-proxy hacia los niveles superiores (el nivel 3 en este caso).

Una vez alcanzado el nivel máximo de tráfico, y a medida que se van incluyendo nuevos niveles a la topología, los requisitos de ancho de banda de las redes locales del tercer nivel se reduce hasta los 1075 Mb/s una vez añadido el 5º nivel, los 840 Mb/s cuando se completa el 6º nivel, para acabar estabilizándose alrededor de los 760 Mb/s con el 7º nivel. El tráfico de las redes locales del resto de niveles de la topología sigue un comportamiento similar al explicado para el tercer nivel.

En la misma figura podemos comprobar que los requisitos de ancho de banda de las redes locales no sobrepasa los 1.100 Mb/s, lo cual demuestra definitivamente la escalabilidad de la arquitectura Proxy-Tree.

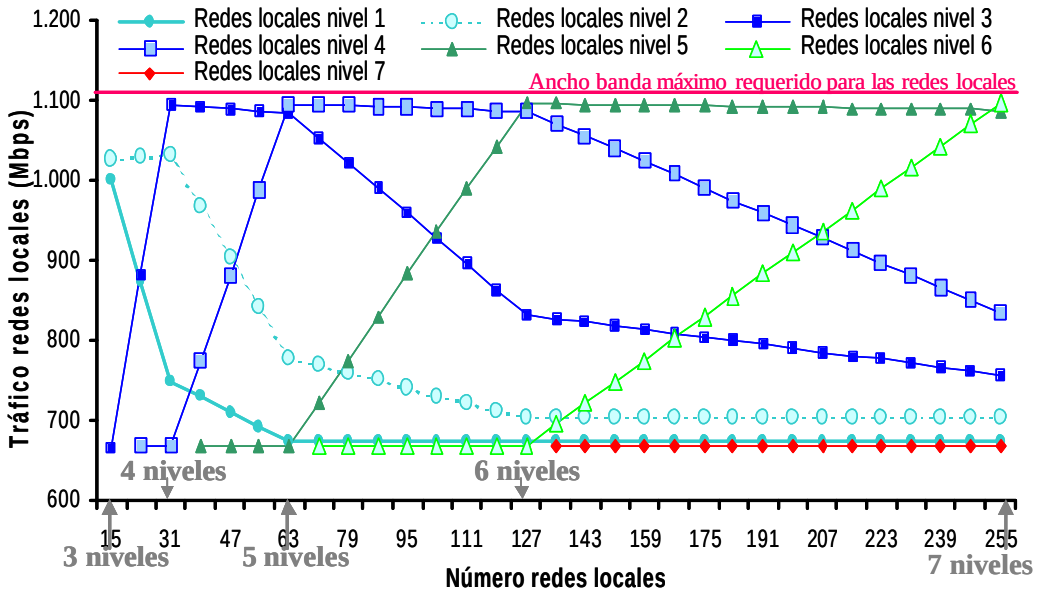


Figura 2-12. Escalabilidad en las redes locales de la arquitectura P-Tree

2.5 Evaluación del rendimiento de la arquitectura Proxy-Tree

Una vez verificada la escalabilidad de la arquitectura P-Tree, es necesario evaluar su rendimiento, comparándolo con otras arquitecturas similares. Este análisis se realizará utilizando el modelo analítico desarrollado en el apartado anterior, en concreto, el ancho de banda efectivo se calculará utilizando la expresión (2.12).

Antes de poder evaluar el rendimiento de la arquitectura P-Tree es necesario ajustar una serie de parámetros para poder optimizar la utilización de sus recursos. Los parámetros principales que hay que tener en cuenta en la arquitectura son el orden del árbol de la topología (binario, terciario, etc.) y la distribución del almacenamiento del proxy entre los esquemas de cache y mirroring.

2.5.1 Análisis del orden del árbol

En la tabla 2-2 mostramos el rendimiento de varios sistemas P-Tree con una capacidad para el 40% de los videos del catálogo y utilizando diferentes órdenes para el árbol de la topología.

Como podemos observar en los resultados, la topología con un árbol binario es la que ofrece unos mejores resultados. La utilización de árboles con un mayor número de hijos implican un peor rendimiento para la arquitectura y un incremento del tráfico recibido por la red principal.

Tabla 2-2. Rendimiento sistemas P-Tree con diferentes topologías en árbol

Sistema P-Tree	Ancho de banda efectivo	Ancho banda red principal
Árbol binario	89.144 Mb/s	75 Mb/s
Árbol terciario	88.791 Mb/s	84 Mb/s
Árbol cuaternario	87.418 Mb/s	90 Mb/s

La razón de este comportamiento la podemos encontrar en la conectividad de las nuevas topologías. La utilización de árboles con un orden mayor implica una mejora de la conectividad en los nodos de los niveles intermedios (pasando de 3 a 4 nodos adyacentes), lo cual conlleva una mayor eficiencia del mirror distribuido. Sin embargo, esta mejor conectividad no es suficiente para compensar el pobre rendimiento de los nodos terminales del árbol (aquellos que no tienen hijos) y que normalmente están situados en el último nivel. Estos nodos son los que tienen una peor conectividad de toda la topología, provocando que un gran número de peticiones se deban servir a distancia-3 ó superiores.

Los árboles terciarios y cuaternarios contienen un mayor número de nodos terminales que los árboles binarios, provocando que el peor rendimiento de estos nodos compense los beneficios obtenidos por la mejor conectividad de los nodos intermedios. Por esta razón, nos hemos decantado por la utilización de árboles binarios en la topología de la arquitectura P-Tree.

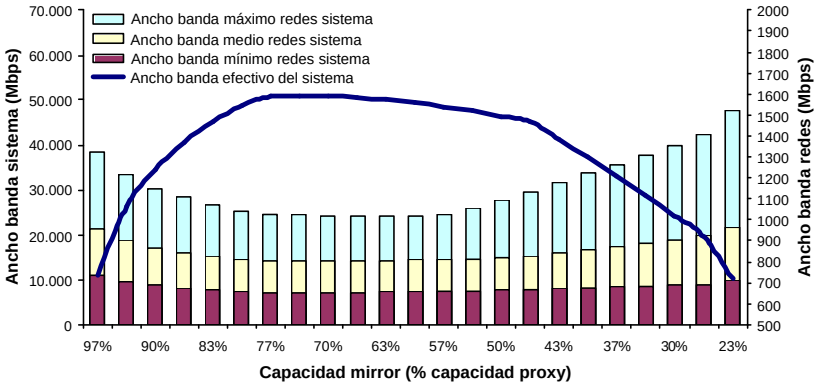
2.5.2 Distribución del almacenamiento de los servidores-proxy

Uno de los parámetros que tiene más incidencia sobre el rendimiento de la arquitectura P-Tree es la distribución de la capacidad del proxy entre los dos esquemas de gestión de contenidos. Una distribución incorrecta puede afectar al rendimiento del sistema. Si no se asigna la suficiente capacidad al esquema de mirroring, puede implicar que las peticiones remotas deban servirse desde servidores-proxy lejanos (requiriendo más recursos de ancho de banda red). Por el contrario, si no se le asigna la suficiente capacidad a la cache se reducirá el número de peticiones que se atienden localmente y consecuentemente se incrementa el número de peticiones remotas.

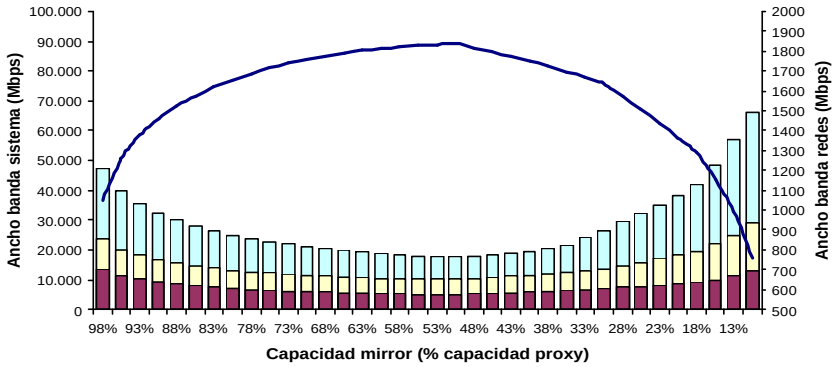
Para estudiar la influencia de este parámetro, hemos analizado el rendimiento obtenido por el sistema a medida que se ha ido variando el parámetro V_c (que especifica la capacidad del proxy dedicada a caching) en la expresión (2.10) que evalúa la probabilidad de acierto en un proxy. Los resultados se han obtenido utilizando un sistema formado por 254 redes locales (7 niveles) y con una capacidad en el proxy para almacenar el 30% y el 40% de las películas. En la figura 2-13 se muestra el ancho de banda efectivo del sistema así como el ancho de banda máximo, medio y mínimo de las redes del sistema. Como se puede observar, cuando se utiliza un proxy con menor capacidad (30%), el mejor resultado se obtiene utilizando un mirror con el 73% del espacio disponible, dejándole a la cache el 27% restante. Al utilizar un proxy mayor (40%) los porcentajes varían, obteniéndose las mejores prestaciones cuando el espacio del proxy se distribuye equitativamente entre los dos esquemas. En ambas configuraciones se obtienen resultados aceptables cuando el espacio dedicado al mirror se encuentra dentro del rango 50%-75%.

2.5.3 Análisis de rendimiento de la arquitectura P-Tree

Una vez seleccionado la distribución óptima de almacenamiento dedicada a cada uno de los esquemas de gestión de los contenidos, ya podemos evaluar el rendimiento de las arquitectura P-Tree.



(a) Capacidad total proxy 30% de los videos del catálogo



(b) Capacidad total proxy 40% de los videos del catálogo

Figura 2-13. Distribución del almacenamiento de los servidores-proxy entre los esquemas de caching y mirroring

En la figura 2-14 se puede observar que las prestaciones que se obtienen para este sistema, entendida como el ancho de banda efectivo del mismo. Para la arquitectura de servidores-proxy de un nivel suponemos que la red principal del sistema en todo momento tiene la suficiente capacidad como para atender todos los fallos de los servidores-proxy. Por lo tanto, el ancho de banda de esta red crece cuando se amplia el sistema y podemos comparar las prestaciones obtenidas por nuestra arquitectura con las obtenidas con un sistema clásico de proxy de un nivel.

A primera vista se puede observar una perdida de prestaciones de la arquitectura P-Tree. La reducción del ancho de banda efectivo del sistema jerárquico con respecto al mismo sistema pero con un nivel de servidores-proxy depende del tamaño de los servidores-proxy. Así, la perdida de rendimiento se puede cuantificar en un 36% (80000 Mb/s versus 51000 Mb/s) para un sistema con 254 redes locales y un proxy del 30%.

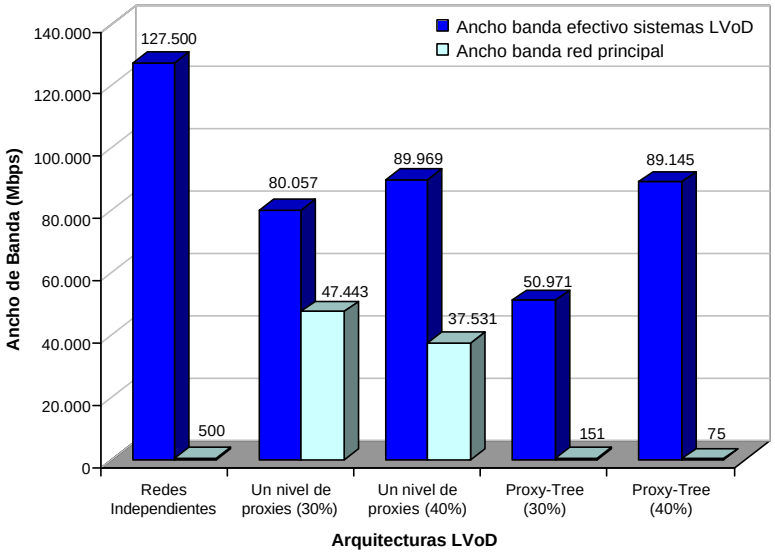


Figura 2-14. Rendimiento de las arquitecturas de LVoD

La diferencia de prestaciones entre ambas arquitecturas se puede justificar por el nivel de distribución de la gestión de ambos sistemas. La arquitectura de servidores-proxy de un nivel únicamente esta compuesta por 2 niveles, el primer nivel compuesto por los distintos servidores-proxy y el segundo por el servidor principal. Estos dos niveles permiten limitar la distancia de servicio máxima de las peticiones a 2 redes (peticiones atendidas desde el servidor principal). De esta forma, se obtiene una arquitectura más centralizada y por lo tanto, más eficiente.

En cambio, la gestión de los sistemas P-Tree esta más distribuida entre los distintos servidores-proxy y la arquitectura puede estar constituida por más niveles. Ambas características provocan que la distancia de servicio pueda ser mayor que la correspondiente de la arquitectura de servidores-proxy de un nivel.

Además, la utilización del esquema de gestión de mirroring reduce la capacidad del esquema de caching lo cual reduce el porcentaje de peticiones que se pueden servir localmente (comparadas respecto a los servidores-proxy de un nivel, que dedican todo su almacenamiento a caching).

Sin embargo, esta reducción de prestaciones es pequeña si aumentamos la capacidad de los servidores-proxy hasta el 40% de las películas del sistema, entonces la diferencia de rendimiento es de apenas un 1% (89.900 Mb/s versus 89.100 Mb/s). Esta reducción en la perdida de prestaciones es debida a que cuando se utilizan servidores-proxy lo suficientemente grandes los fallos no van más allá de los servidores-proxy situados a distancia uno (al igual que ocurre en los sistemas de servidores-proxy de un nivel), mejorando su rendimiento.

Los resultados obtenidos por la arquitectura de servidores independientes identifica el rendimiento óptimo (127.000 Mb/s) para un sistema en el cual, al igual que ocurre con los sistemas centralizados, todas las peticiones se pueden servir localmente. Comparando los resultados obtenidos mediante la arquitectura de servidores independientes y las arquitecturas con los contenidos distribuidos se puede comprobar que estas últimas ofrecen una menor efectividad. Esta reducción en el rendimiento de las arquitecturas distribuidas respecto a las centralizadas ó de servidores independientes es exclusivamente achacable a los recursos de red requeridos por los sistemas distribuidos debido a las peticiones que no se pueden servir localmente (mayor distancia de servicio).

2.5.4 Requerimientos de ancho de banda de red

Otro factor importante a la hora de diseñar un sistema como el propuesto es el tamaño que se tiene que utilizar para las redes del sistema. El ancho de banda requerido por las redes de un determinado nivel (figura 2-15) depende de la posición relativa de ese nivel en la jerarquía, la cual no es estática sino que se modifica (al igual que sus requerimientos de ancho de banda) a medida que se añaden nuevos niveles al sistema.

Si deseamos garantizar que las redes de un nivel puedan soportar la carga asociada a cualquier nivel de la topología, entonces cuando se amplíe el sistema tenemos que asumir el peor caso y utilizar redes locales con el ancho de banda máximo requerido en cualquiera de los niveles de la topología.

La carga máxima que tiene que soportar una red local del sistema se localiza siempre en el penúltimo y antepenúltimo nivel del árbol (peor caso). Ésta sobrecarga es debida a la cercanía

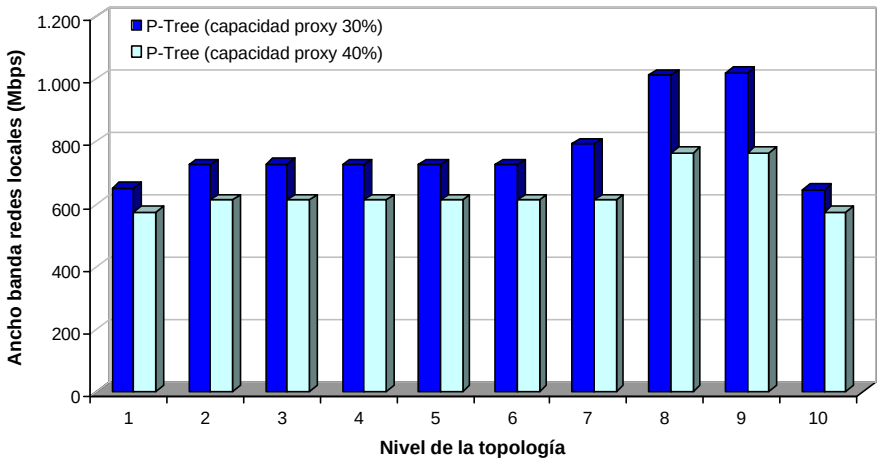


Figura 2-15. Requerimientos de las redes locales por niveles

del último nivel de la topología. Los nodos del último nivel al no disponer de hijos, redirigen todas las peticiones no servidas localmente hacia los niveles superiores, incrementando el tráfico gestionado por las redes locales de esos niveles. La menor conectividad de los nodos del último nivel reducen la eficiencia de los mirrors distribuidos asociados con estos servidores-proxy y provocan la aparición de peticiones que se deban servir a distancia 3 ó incluso a distancia 4.

En el resto de niveles intermedios, los fallos en los servidores-proxy se pueden distribuir entre los servidores-proxy de los niveles superiores e inferiores. A partir de ese nivel el ancho de banda requerido se reduce y se estabiliza.

El ancho de banda máximo requerido en las redes locales en cualquiera de los niveles de la topología es el que se tendrá que utilizar para las redes locales que se añadan al sistema cuando éste se amplié. Para los sistemas que hemos estudiado este ancho de banda es de 760 Mb/s cuando se utiliza un proxy de tamaño 40% y de 1021 Mb/s cuando se utiliza un proxy de tamaño 30%. Nosotros confiamos que este pico se pueda reducir modificando el porcentaje entre caching y mirroring de los últimos niveles y por lo tanto, el ancho de banda de las redes locales se acerque más a la media de las redes locales del sistema: 650 Mb/s (40%) y 800 Mb/s (30%) respectivamente.

2.5.5 Comparación de resultados entre diferentes arquitecturas LVoD

Por último, en la tabla 2-3 resumimos los principales parámetros de un sistema de LVoD, utilizando las siguientes arquitecturas: centralizada, servidores independientes, un nivel de servidores-proxy y Proxy-Tree. El cálculo del número máximo de usuarios independientes que soporta el sistema se ha realizado suponiendo la utilización de políticas de servicio unicast y la utilización de los contenidos multimedia con el formato MPEG-1 y una duración aproximada de 90 minutos, por lo tanto para cada video se requiere un ancho de banda 1.5 Mb/s y almacenamiento de aproximadamente 1GByte.

De los resultados obtenidos podemos deducir que de las principales arquitecturas escalables (sin tener en cuenta los sistemas centralizados), la que ofrece los mejores resultados es la de servidores independientes, pero a costa de incrementar el volumen de almacenamiento requerido hasta alcanzar los 25 Terabytes.

Las dos arquitecturas basadas en servidores-proxy, obtienen resultados parecidos (90.000 Mb/s y 89.000 Mb/s, para el sistema de servidores-proxy de un nivel y P-Tree respectivamente). La arquitectura de un nivel de servidores-proxy se caracteriza por requerir todavía un servidor central complejo y una red principal con un gran ancho de banda (más de 37.000 Mb/s).

Mientras, la arquitectura P-Tree permite una **mayor escalabilidad** y únicamente requiere la utilización de **redes locales y servidores-proxy de menor complejidad**.

Tabla 2-3. Principales parámetros de las distintas arquitecturas LVoD

Parámetros	Centralizado	Servidores Independientes	Un nivel de servidores-proxy	Proxy-Tree
Redes locales	0	0	254	254
Redes principales	1	255	1	1
Servidores	1	255	1	1
Servidores-proxy	0	0	254	254
Capacidad Servidores-proxy	0	0	40%	40%
Almacenamiento requerido *(100 videos x 1GB)*	100 GBytes	25.500 GBytes	10.200 GBytes	10.200 GBytes
Ancho banda de la red principal	127.500 Mb/s	0 Mb/s	37.500 Mb/s	75 Mb/s
Ancho banda redes locales	0 Mb/s	500 Mb/s	500 Mb/s	650 Mb/s (media)
Ancho de banda efectivo	127.500 Mb/s	127.500 Mb/s	90.000 Mb/s	89.000 Mb/s
Número máximo usuarios soportados	90.000	90.000	60.000	59.333
Escalabilidad	limitada	ilimitada	limitada	ilimitada

Con respecto a la arquitectura de servidores independientes el sistema P-Tree permite una **mayor tolerancia a fallos** del sistema, un mejor **balanceo de la carga** entre los distintos servidores (si un servidor esta saturado, puede redirigir sus peticiones hacia los servidores-proxy vecinos) y permite un **mayor potencial de efectividad de las políticas multicast** (el número de usuarios que pueden acceder a un determinado contenido es del orden de 4 veces superior comparado con las arquitecturas de servidores independientes).

2.6 Conclusiones

En este capítulo hemos propuesto una topología jerárquica en árbol basada en la utilización de redes locales independientes. Estas redes utilizarán un proxy local para poder servir las peticiones a los videos más populares sin necesidad de recurrir al servidor principal. En caso de fallo del proxy no se accede al servidor directamente sino que se intenta servir la petición desde las redes locales más cercanas en la topología.

Para mejorar la escalabilidad del sistema se ha modificado la funcionalidad del proxy de forma que funcione al mismo tiempo como cache para las películas más vistas y como mirror del resto de películas. La arquitectura Proxy-Tree garantiza un crecimiento ilimitado y a bajo coste del sistema de VoD. Además la capacidad del sistema se puede adaptar fácilmente a cualquier número de usuarios y, por lo tanto, no requiere su sobredimensionamiento inicial para poder permitir un posterior crecimiento. La arquitectura propuesta permite escalabilidad tanto vertical (aumentando la capacidad de servicio del servidor ó aumentado el almacenamiento) como escalabilidad horizontal (incluyendo nuevas redes).

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. Además, al distribuir el servicio de las peticiones entre los servidores-proxy, el tamaño del servidor requerido (y su coste) es mucho menor que en una sistema parcial ó totalmente centralizado.

Se ha desarrollado un modelo analítico para evaluar la capacidad de servicio y la escalabilidad (dos de los parámetros más importantes de las sistemas LVoD) tanto de la arquitectura propuesta como de las otras alternativas presentes en la literatura para la implementación de un sistema LVoD. Mediante este modelo, no solo podemos calcular el rendimiento del sistema (ancho de banda efectivo) sino que también nos permite evaluar como evolucionan lo requisitos de los distintos componentes del sistema (redes y servidores) a medida que el sistema crece.

A partir de los resultados proporcionados por el modelo analítico hemos podido confirmar la escalabilidad ilimitada de la arquitectura propuesta. Además se ha podido constatar que el ancho de banda de la red requerido por la topología P-Tree es ligeramente mayor que la arquitectura basada en servidores-proxy, aunque esto no implica necesariamente un coste mayor.

El sistema de servidores-proxy de un nivel requiere una red principal con un ancho de banda considerablemente más grande que el del sistema P-Tree (unos 3 órdenes de magnitud más grande). El coste de una red no crece linealmente con el ancho de banda, sino que a medida que se incrementa el nivel de prestaciones (ancho de banda) exigido a la red, el coste es cada vez mayor. Por lo tanto, no es irrazonable suponer que el coste final de nuestra arquitectura será inferior al coste de la arquitectura de servidores-proxy de un nivel, principalmente por culpa del sobre coste requerido para la red y el servidor principal.

Por otro lado, al igual que en la mayoría de los sistemas distribuidos, nuestra arquitectura tiene que sacrificar una parte de su eficiencia (con respecto al resto de arquitecturas LVoD

centralizadas ó con alta replicación de contenidos) para soportar la tolerancia a fallos y la escalabilidad.

En el siguiente capítulo analizaremos el rendimiento de la arquitectura P-Tree con el objeto de reducir su penalización con respecto a las prestaciones de arquitecturas similares.