UNIVERSIDAD COMPLUTENSE DE MADRID

FACULTAD DE INFORMÁTICA
Departamento de Arquitectura de Computadores y Automática

UN SERVIDOR HÍBRIDO ADAPTATIVO PARA LA DIFUSIÓN DE INFORMACIÓN EN ENTORNOS DE COMUNICACIÓN ASIMÉTRICOS CON RESTRICCIONES TEMPORALES.

MEMORIA PARA OPTAR AL GRADO DE DOCTOR

PRESENTADA POR

Jesús Fernández Conde

Bajo la dirección del doctor

Daniel Mozos Muñoz

Madrid, 2011

ISBN: 978-84-694-2881-8

Un Servidor Híbrido Adaptativo para Difusión de Información en Entornos de Comunicación Asimétricos con Restricciones Temporales

---

Jesús Fernández Conde

Tesis Doctoral

Universidad Complutense de Madrid

Dpto. Arquitectura de Computadores y Automática

Un Servidor Híbrido Adaptativo para Difusión de Información en Entornos de Comunicación Asimétricos con Restricciones Temporales

---

Memoria presentada por Jesús Fernández Conde para optar al grado de doctor por la Universidad Complutense de Madrid, realizada bajo la dirección de D. Daniel Mozos Muñoz.

Madrid, Septiembre 2010

Este trabajo ha sido posible gracias a la financiación de la Comisión Interministerial de Ciencia y Tecnología, a través de los proyectos CICYT TEC2005-04752, TIN2006-03274 y TIN2009-09806.

Agradecimientos

El camino ha sido largo y duro, lleno de altibajos, no me acabo de creer que esté ahora mismo escribiendo esta página de agradecimientos, ya que suele ser la última en escribirse y significa que la memoria de la tesis está próxima a su versión definitiva.

Por supuesto quiero agradecer lo primero de todo el apoyo de familiares y amigos. Aunque no hayan podido ayudarme en cuestiones técnicas, siempre me han dado mucho ánimo, que es lo más importante. Sin su presencia me hubiese sido imposible llevar a cabo no sólo este trabajo sino la mayoría de mis logros. Primo Jesús, por fin puedo decir con orgullo que te debo una cena...

Mis compañeros del despacho INF-347 me han hecho pasar muy buenos momentos. Mención especial merece Ángel, mi mejor compañero de todo tipo de batallas, siempre dispuesto para la acción. También el señor Guillermo, qué bueno aquel congreso en Croacia. No me quiero olvidar del doctor Poletti, con sus comentarios que no alteran la entropía de mi universo. Y por supuesto las últimas incorporaciones Carlos, Juanan, Pablo y Alberto, vosotros sí que llegaréis lejos.

Pasando al plano técnico, debo dar las gracias a toda la gente que ha puesto de forma directa o indirecta su granito de arena. En especial, gracias a mis compañeros de UMass Oscar y Ping Xuan, con vosotros empezó toda esta historia allá por el 97. También agradecer el tiempo que me dedicaba mi *advisor* en UMass, Krithi Ramamritham,

a pesar de su permanente sobresaturación. Qué buenos tiempos aquellos...

Por último, y no por ello menos importante, quiero agradecerle a Daniel la paciencia y comprensión casi infinitas que ha demostrado poseer. Pienso de verdad que con cualquier otro tutor esta tesis no hubiera llegado nunca a buen término.

Capítulo 1: Introducción

1.1 Introducción

La informática para dispositivos móviles (en inglés ***mobile computing***) ha evolucionado de manera vertiginosa gracias a la convergencia de dos tecnologías bien diferenciadas: la aparición de potentes dispositivos móviles y el desarrollo de redes sin cables rápidas y fiables. Hace menos de dos décadas era muy difícil imaginar un número masivo de equipos móviles efectuando peticiones de información a bases de datos utilizando redes sin cables.

Hoy este escenario se ha convertido en una realidad perfectamente asumida, y nos encontramos con millones de personas que llevan consigo a diario dispositivos tales como PDAs (*Personal Digital Assistants*), ordenadores portátiles, teléfonos móviles de última generación, etc., con capacidad para acceder a una gran variedad de información desde cualquier lugar y en cualquier momento. La meta del *ubiquituous computing* [Weis91] está cada vez más cercana.

A finales de 2008 Europa contaba con 14,7 millones de conexiones móviles de banda ancha. Esa cifra no sólo supone el 74% de incremento respecto al año anterior, sino que ya constituye el 11,6% del total de conexiones de banda ancha europeas. Está previsto que los accesos móviles a Internet en Europa sigan aumentando a un ritmo del 30% hasta alcanzar los 70 millones en el año 2014.

Según datos de la Comisión del Mercado de las Telecomunicaciones (www.cmt.es), la telefonía móvil en España añadió 229.903 nuevas altas en enero de 2009 hasta superar los 51,12 millones de líneas, cifra que supone un aumento del 3,1% respecto al mismo periodo del año anterior. Dado que la población censada se sitúa en España en torno a los 46 millones de habitantes, el número de líneas por persona es aproximadamente de 1,1.

A pesar de la actual coyuntura económica negativa, el segmento de los teléfonos avanzados o *smartphones* continúa en expansión, tanto en términos absolutos (36,4 millones de unidades vendidas en 2009, el 12,7% más que el año anterior) como

relativos (el 13,9% de los móviles vendidos en 2009 fueron *smartphones*, mientras que en 2008 eran solamente el 11%).

La figura 1.1 representa la arquitectura de un sistema de comunicación para dispositivos móviles, la cual incluye una serie de elementos fijos (estaciones base o de soporte a móviles), los cuales incorporan interfaces para la comunicación con los propios dispositivos móviles que se encuentran dentro de un determinado área de cobertura llamada celda, representada habitualmente de forma hexagonal.

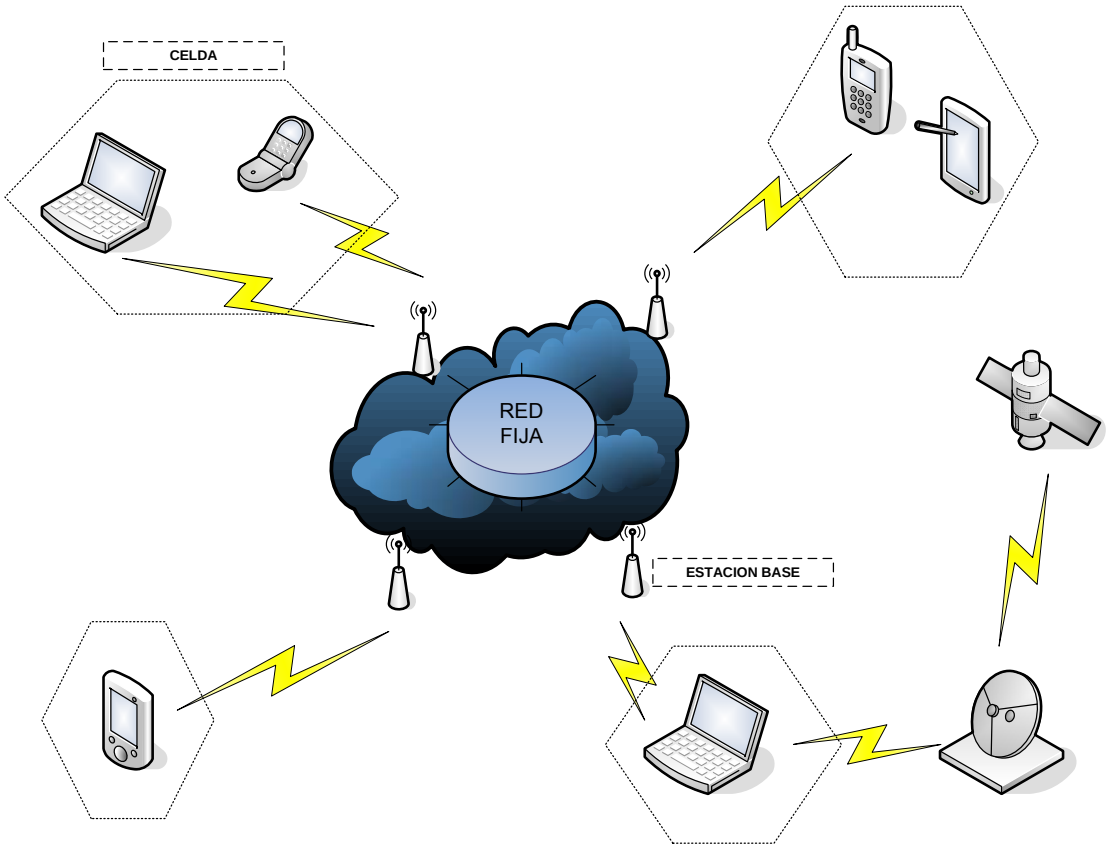

Figura 1.1. Arquitectura de sistema de comunicación para dispositivos móviles

El objetivo fundamental de este tipo de sistemas es ofrecer acceso teóricamente ilimitado a todo tipo de información almacenada en los servidores de la parte fija de la red. Sin embargo, las limitaciones de ancho de banda de las redes sin cables, la relativamente corta vida activa de las baterías de los dispositivos móviles y la movilidad de los usuarios hacen el problema del acceso a la información más complicado que el equivalente en redes diseñadas para usuarios no móviles.

En resumen, en estos sistemas de comunicación, existe una clara necesidad de servidores de información con capacidad para proporcionar diferentes tipos de información (informes sobre el estado del tráfico, tiempo, noticias, bolsa, etc.) a un número potencialmente ilimitado de usuarios móviles. El diseño de servidores de información eficientes y escalables con una utilización mínima de recursos necesarios es una tarea harto complicada, debido principalmente a las restricciones en el uso del ancho de banda compartido y a la movilidad de los usuarios.

En comparación con la transmisión punto a punto, en la cual se establece un canal lógico entre cada usuario y el servidor, la transmisión de información empleando difusión (en inglés se utiliza el término ***broadcast***) resulta ser mucho más atractiva en estas situaciones, ya que la información es recibida simultáneamente por todos los usuarios situados en la zona de servicio del servidor, proporcionando la potencialidad para hacer un uso eficaz del limitado ancho de banda disponible.

La eficacia de la difusión aumenta en servicios basados en localización (por ejemplo, información sobre vuelos en un aeropuerto), ya que los usuarios que se encuentran próximos tanto

espacial como temporalmente tienden a solicitar información con un alto grado de correlación.

La tecnología de difusión ha sido utilizada desde hace muchos años para proporcionar información a un gran número de usuarios [Wong88]. Podemos pensar en los sistemas de radio y televisión, en los que un número ilimitado de usuarios tienen acceso a programas en base a una planificación preprogramada.

En los sistemas de comunicación actuales para usuarios móviles, en la inmensa mayoría de los casos, los usuarios requieren que la información les llegue cumpliendo ciertos plazos temporales. Por ejemplo, consideremos un servidor de información de tráfico, y un usuario que conduce un vehículo y debe tomar una decisión sobre dos alternativas posibles para llegar a su destino (ver figura 1.2).

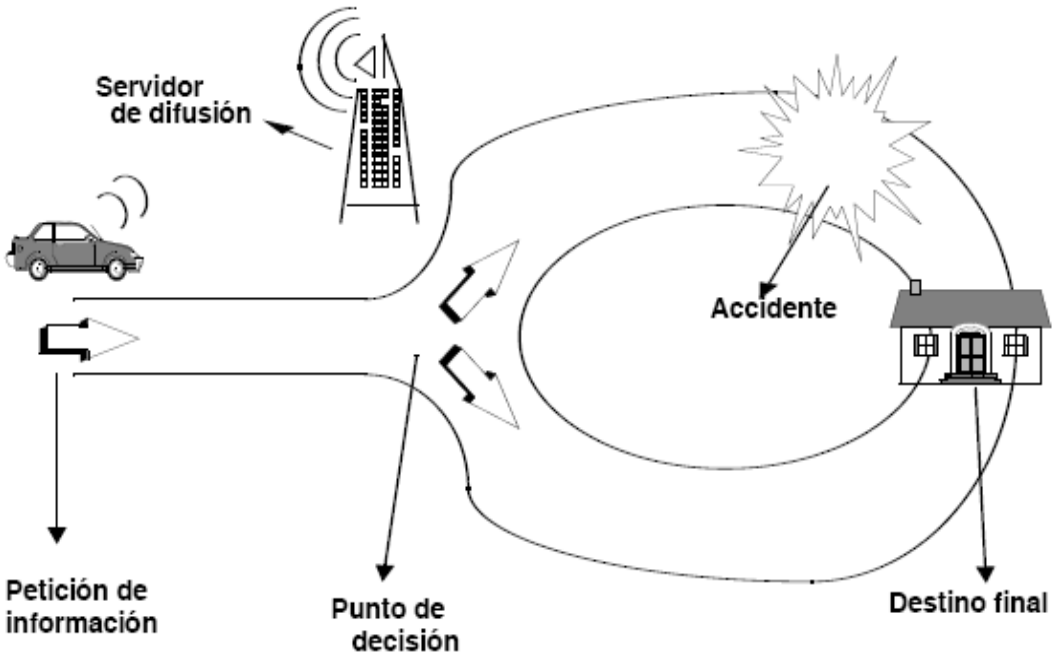

Figura 1.2. Entorno de Comunicación Asimétrico con Restricciones Temporales

Obviamente, el servidor debe proporcionar el estado del tráfico al usuario (por ejemplo, una ruta puede estar congestionada debido a un accidente) antes de que el conductor alcance el punto de decisión, de lo contrario la información carecerá de valor alguno para el usuario.

Además, teniendo en cuenta que en este tipo de entornos el volumen de información transmitido por el servidor es mucho mayor que el transmitido por los usuarios, nos vamos a referir a estos escenarios como ***entornos de comunicación asimétricos con restricciones temporales***. La asimetría de la comunicación es debida a varios factores, principalmente asimetría en las aplicaciones, los equipos y la red de comunicación.

Esta tesis presenta un modelo de servidor de difusión híbrido adaptativo, con las características necesarias para funcionar de manera efectiva en entornos de comunicación asimétricos con restricciones temporales. El modelo presentado extiende la funcionalidad y prestaciones de modelos anteriores por medio de un algoritmo de planificación adaptativo que tiene en cuenta la frecuencia de acceso a la información, las limitaciones de ancho de banda en los canales de bajada y de subida, y las restricciones temporales asociadas a las peticiones.

Con el objetivo de minimizar el número total de plazos perdidos, y en consecuencia satisfacer al mayor número posible de usuarios, el servidor adapta dinámicamente la información servida periódicamente. Esta adaptación de la *cantidad* y el *contenido* de la información diseminada periódicamente resulta ser muy efectiva frente a perfiles de usuario cambiantes y restricciones temporales asociadas a los datos.

1.2 Objetivos de la tesis

Los principales objetivos de la presente tesis son la presentación, estudio, análisis y evaluación cuantitativa de un modelo de servidor de difusión híbrido adaptativo que se comporte de manera eficiente en entornos de comunicación asimétricos con restricciones temporales y perfiles dinámicos de acceso a la información.

Nuestra propuesta se basa en el aprovechamiento máximo del ancho de banda disponible con el objetivo de minimizar el número de plazos perdidos y en consecuencia satisfacer el mayor número de peticiones de información.

1.3 Estructura de la memoria

La memoria de la tesis está estructurada de la forma siguiente:

La sección 2 detalla el estado del arte, presentando las ideas fundamentales acerca de los modelos de transmisión de información, en concreto los sistemas de diseminación de información que utilizan difusión. Se incluye un análisis comparativo de modelos de transmisión de información que utilizan difusión.

La sección 3 expone las principales consideraciones a tener en cuenta en el diseño de servidores de información híbridos adaptativos en entornos de comunicación asimétricos con restricciones temporales y perfiles de acceso dinámicos.

En la sección 4 se realiza un análisis teórico del número mínimo de peticiones no satisfechas en plazo en un servidor de difusión para una determinada distribución de acceso a la información en un entorno de comunicación asimétrico con restricciones temporales.

Nuestro modelo de servidor híbrido adaptativo es presentado en la sección 5, en la cual detallamos los algoritmos de planificación empleados por el mismo tanto para la selección de la información servida de forma periódica como para la difusión de información bajo demanda. También se detalla la técnica empleada para la estimación de las frecuencias de acceso a la información.

La sección 6 presenta y analiza los resultados experimentales obtenidos por medio de simulación. Hemos simulado un entorno de comunicación asimétrico que consta de un único servidor y un número ilimitado de usuarios móviles que pueden encontrarse en la zona de influencia del servidor durante un determinado periodo temporal.

La sección 7 resume y concluye el trabajo, enumerando las líneas de trabajo futuro.

Finalmente, la sección 8 cita los artículos publicados por el autor en relación con el presente trabajo de investigación.

Capítulo 2:
Estado del arte

2.1 Introducción

En este capítulo comenzamos presentando las ideas fundamentales acerca de los modelos de transmisión de información, centrando nuestra atención en sistemas de diseminación de información mediante difusión. Seguidamente realizamos un análisis comparativo de modelos de transmisión de información que utilizan difusión.

2.2 Transmisión de información

Los constantes avances en el terreno de las comunicaciones incluyendo la proliferación de Internet, el intenso desarrollo de redes inalámbricas para usuarios móviles y la creciente disponibilidad de enlaces de banda ancha para usuarios finales han impulsado el desarrollo de un amplio rango de aplicaciones de acceso a todo tipo de información.

La evolución de las tecnologías de redes sin cables y dispositivos móviles, tales como GSM/GPRS/UMTS, *wireless* LAN o Bluetooth, han revolucionado la forma en que nos comunicamos e intercambiamos información, haciendo que tanto la comunicación vocal como el acceso a redes de datos estén disponibles en cualquier instante y prácticamente desde cualquier lugar. Todas las tecnologías para redes sin cables continúan evolucionando impulsadas por la demanda del mercado para conseguir mayores velocidades de transmisión de datos.

La computación móvil es hoy algo totalmente habitual, y es muy frecuente que cualquier persona en un país desarrollado lleve consigo al menos un dispositivo portátil de forma permanente. También es cada vez más común la necesidad de acceso a servicios de información a través de una infraestructura compartida, sin importar la localización física o la movilidad de los usuarios.

Los sistemas y servicios de datos para usuarios móviles representan un segmento en constante crecimiento y de creciente importancia en la industria de las comunicaciones. Por dar una cifra, el número de usuarios de telefonía móvil GSM superaba los 1500

millones en el año 2006 [Saut06], algo impactante si tenemos en cuenta que la tecnología GSM apareció a principios de los 90.

Los usuarios de estos sistemas de información demandan servicios como correo electrónico, acceso remoto a ordenadores personales y redes de oficina, acceso a información de todo tipo, acceso a Internet, etc. Existen otros servicios más específicos para empresas como gestión de flotas, emergencias, atención al cliente, por citar algunos.

En resumen, existen dos factores principales que motivan que los servicios de datos para usuarios móviles estén en plena expansión. El primero es el incremento exponencial en el uso de dispositivos como teléfonos móviles, ordenadores portátiles y PDAs, cada vez más baratos y más fiables. El segundo es la necesidad creciente de Internet para todo tipo de operaciones en la vida diaria.

En estos entornos de comunicación surgen nuevos retos en el acceso a la información, ya que tanto la localización como el número de usuarios varían con el tiempo. Además existen restricciones añadidas como el ancho de banda limitado (mucho más acusado en redes sin cables) y limitaciones de consumo.

Según [Fran96], los diferentes mecanismos para transferencia de información pueden ser clasificados de acuerdo a tres características fundamentales:

1. Transferencia de información iniciada por la fuente de información o por el destinatario. Los servidores de bases de datos actuales y los repositorios de objetos manejan información para clientes que solicitan datos de forma

explícita cuando los necesitan. Cuando se recibe una solicitud en el servidor, éste localiza la información de interés y se la envía al cliente. Este estilo de operación solicitud-respuesta se denomina *client-pull*, ya que la transferencia de información es iniciada por el cliente. En contraste, existen otros sistemas en los que la información es enviada por el servidor sin petición explícita, llamados *server-push*.

2. Transferencia de información periódica (basada en un programa que se repite regularmente) o aperiódica. La transferencia de información periódica se realiza en base a la repetición regular de un programa (por ejemplo, una aplicación que envía información bursátil periódicamente), mientras que en la aperiódica no existe un programa periódico (por ejemplo, una aplicación que envía información bursátil cada vez que se produce un cambio significativo).

3. Transferencia de información punto a punto (una sola fuente y un solo destinatario) o punto a multipunto (una sola fuente y múltiples destinatarios).

Vamos a centrarnos en este tercer punto. En las transferencias de información punto a punto, también llamadas unidifusión (*unicast*), se establece un canal lógico entre la fuente y el destinatario, por lo que si se desea enviar la misma información a *n* destinatarios, habrá *n* comunicaciones punto a punto independientes o dicho de otro modo *n* copias de la misma información enviadas desde el origen.

Por ejemplo, en la figura 2.1 la máquina M1 envía una copia de la información a M2 y otra copia a M4.

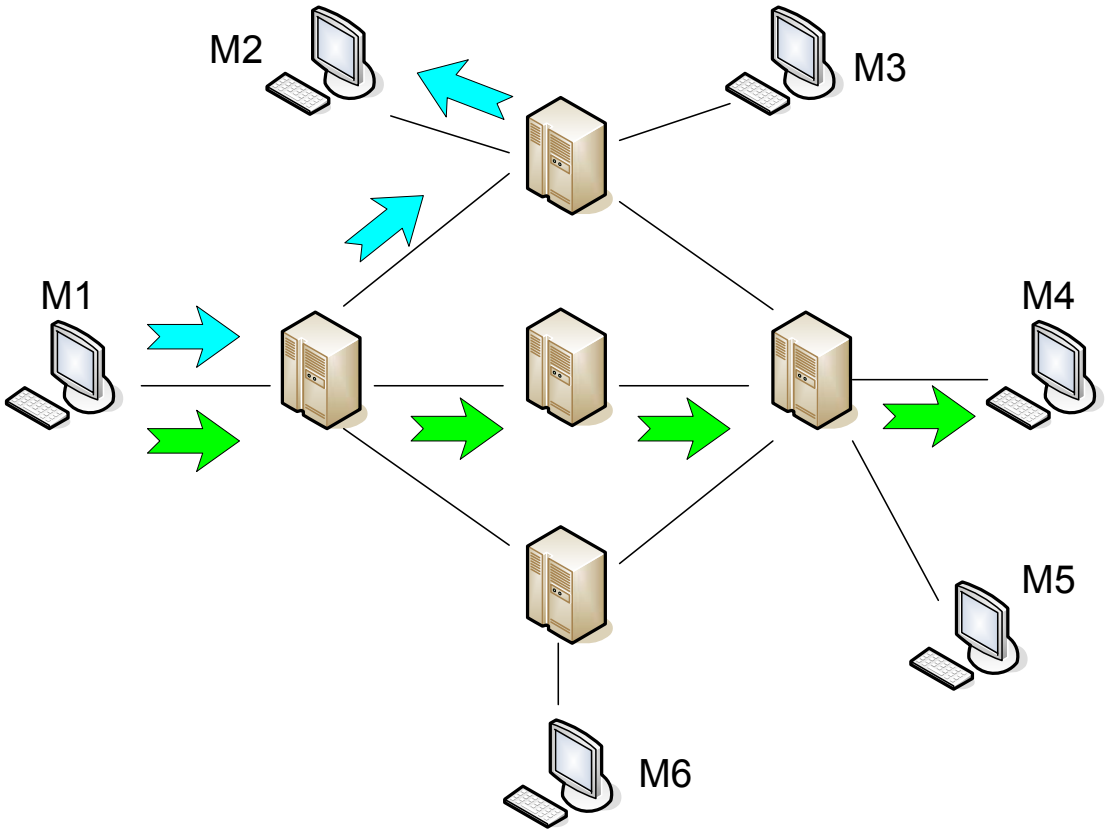

Figura 2.1. Unidifusión

Dentro de las transferencias punto-a-multipunto nos encontramos con dos opciones: multidifusión (*multicast*) y difusión (*broadcast*).

La multidifusión se basa en un único proceso de envío de la misma información, independientemente del número de receptores, a todos los receptores que sean miembros de un determinado grupo de multidifusión. La información sólo se envía una vez, no se transmiten *n* copias. La información se envía a un conjunto de destinatarios conocidos a priori. En la figura 2.2 la máquina M1 envía una sola copia de la información a M2 y M4, que pertenecen al mismo grupo de multidifusión (Grupo 1).

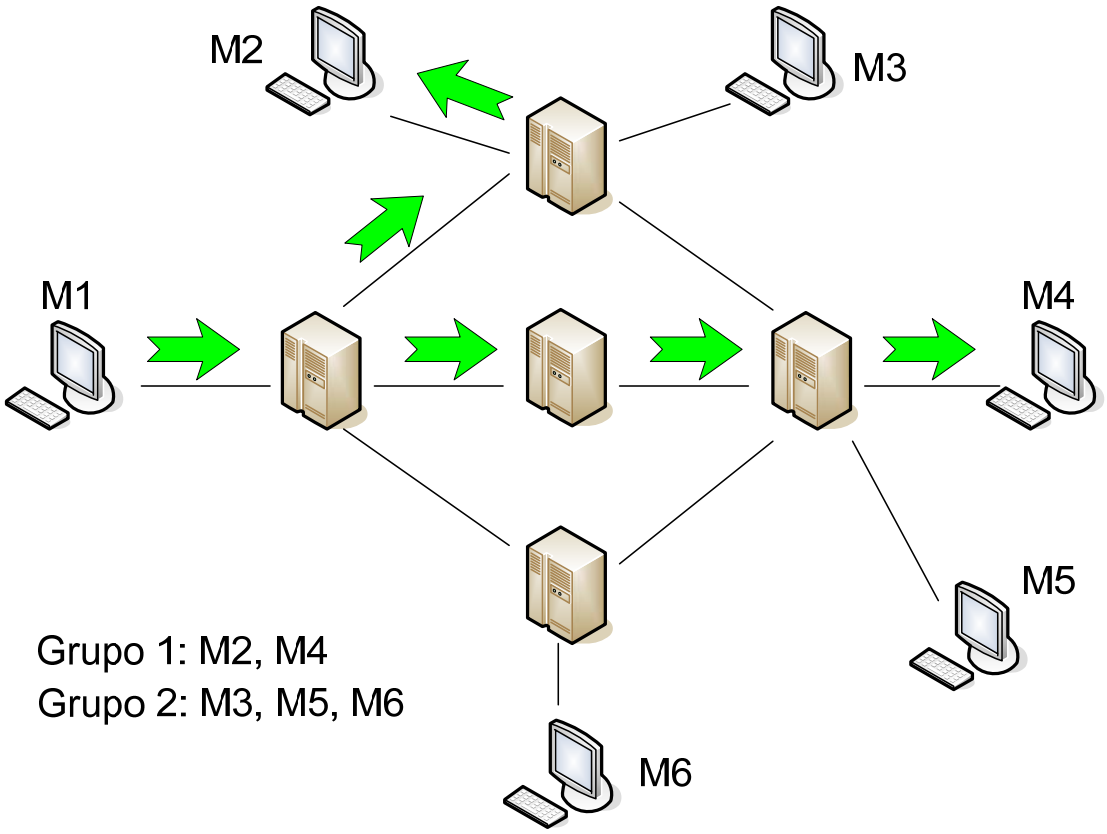

Figura 2.2. Multidifusión

En difusión, el conjunto de destinatarios que recibe la información son todos los potenciales usuarios que se encuentren en la misma red o área de influencia del transmisor, siendo por tanto un conjunto desconocido y no identificado. En la figura 2.3 la máquina M1 transmite una única copia de la información a todas las máquinas conectadas a la misma red de comunicaciones.

En comparación con la transmisión de información punto a punto, la tecnología de difusión tiene el potencial para aprovechar más el ancho de banda disponible para la diseminación de grandes volúmenes de información a un gran número de usuarios, ya que

puede satisfacer a un número ilimitado de clientes con una única transmisión.

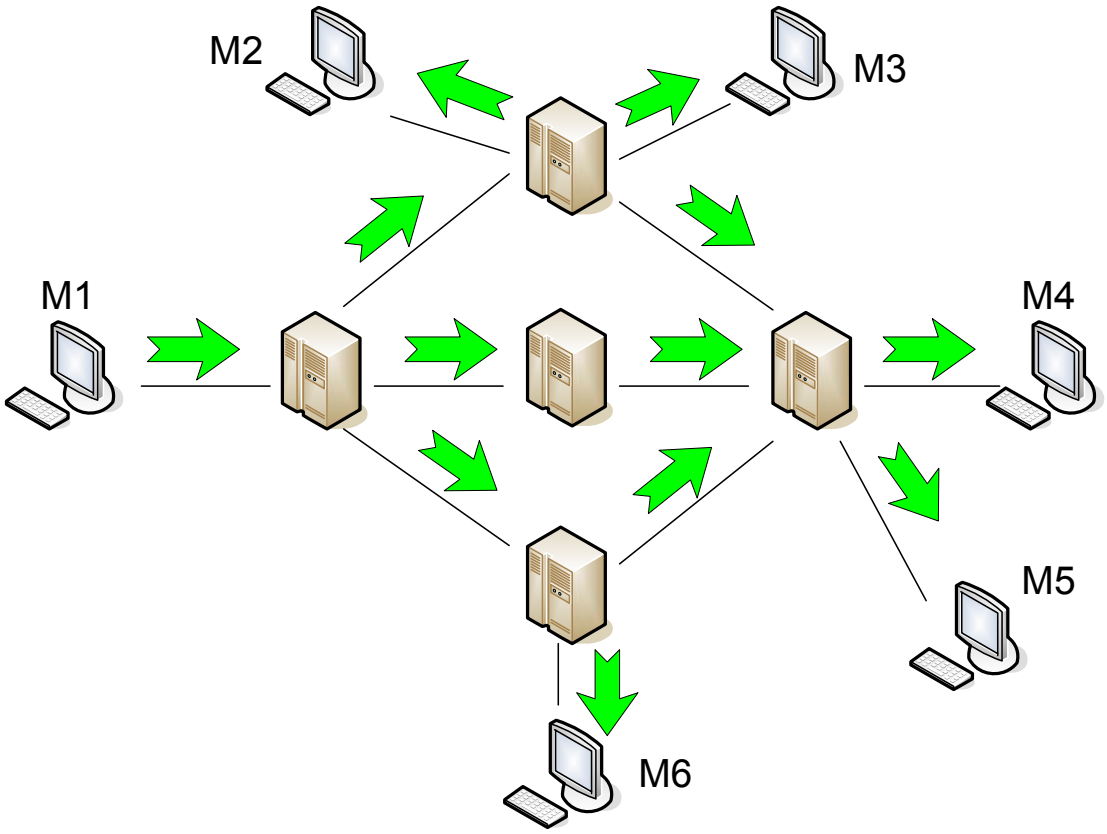

Figura 2.3. Difusión

En este sentido, la difusión es particularmente apropiada para entornos de comunicación para dispositivos móviles. En dichos entornos, los sistemas punto a punto son inadecuados por falta de recursos en cuanto a ancho de banda y capacidad de proceso del servidor. La multidifusión tiene utilidad limitada ya que es necesario que el conjunto de usuarios que reciben la información pertenezca a un grupo definido de antemano.

2.3 Diseminación de información mediante difusión

Como ya hemos mencionado, cuando se utiliza difusión, la información es enviada por el servidor simultáneamente a todos los clientes en su área de influencia. Los clientes tienen toda la información disponible en el medio de comunicación y seleccionan aquella información que les es de utilidad. La difusión es una tecnología que se ha utilizado tradicionalmente en sistemas de radio y televisión para diseminación de información, en base a una planificación temporal sabida de antemano.

En [Wong88] se detallaban arquitecturas y prestaciones de sistemas que utilizaban un canal de difusión para transmitir información a una comunidad de usuarios. La información se organizaba en unidades de información llamadas páginas. Ya se mencionaba la necesidad de algoritmos de planificación eficientes para minimizar el tiempo de espera de los usuarios.

En cuanto a soporte físico de difusión, la difusión por satélite puede ser usada como un enlace de bajada de gran capacidad para servir información a usuarios. La difusión puede utilizarse también en redes cableadas, por ejemplo Ethernet o cualquier otra red de área local. Existen muchas infraestructuras proporcionadas por tecnología de cable y satélite que soportan difusión [Hel99].

Aunque estas tecnologías varían en muchos aspectos, podemos pensar en un modelo abstracto donde los usuarios acceden a la información a través de puntos de acceso. El modelo abstracto consiste en una estación base o servidor, un número en

principio ilimitado de usuarios o clientes y al menos un canal de comunicación en sentido servidor-clientes (llamado comúnmente canal de bajada) a través del cual el servidor envía la información.

Suele existir también un canal de comunicación en sentido contrario (llamado comúnmente canal de subida), utilizado por los usuarios para enviar peticiones a la estación base. Es asumido que todos los usuarios pueden recibir la información de forma simultánea, mientras que el canal de subida sólo soporta una transmisión de un cliente en un instante dado.

Existen multitud de aplicaciones en los que la difusión puede ser utilizada, como por ejemplo comercio electrónico, sistemas de información de tráfico, televisión bajo demanda, información de bolsa, sistemas de distribución de noticias, etc. La difusión aprovecha la gran capacidad del canal de bajada (frente al canal de subida), satisfaciendo una gran cantidad de peticiones de elementos de información muy populares con una sola respuesta.

Existen sistemas comerciales, como por ejemplo AirMedia (www.airmedia.co.za) o Hughes DirecPC (www.direcpc.com) que permiten la diseminación de la información disponible en Internet a usuarios móviles. En el sistema DirecPC el servidor transmite información a todos los clientes a través de un canal de bajada vía satélite, mientras que las peticiones de los usuarios se envían a través de un acceso telefónico de baja capacidad.

En los sistemas de intercambio de información tradicionales, siempre se ha asumido que el medio de comunicación físico era simétrico, esto es, el ancho de banda disponible para recibir datos de la red era igual al disponible para enviarlos. Esto era lógico ya

que típicamente se compartía el mismo medio físico (cable) para la comunicación en ambos sentidos.

Los sistemas basados en difusión son inherentemente entornos de comunicación asimétricos. La asimetría no sólo se produce en el ancho de banda disponible (mucho mayor en bajada que en subida) sino también en el volumen de datos intercambiados (mucho mayor en bajada), y número de clientes frente a número de servidores. Esto no ocurre solamente en modelos de servicio para redes sin cables, sino también en modelos de servicio para redes fijas, por ejemplo en ADSL. Podemos citar como ejemplo una red de distribución de vídeos por Internet, en la que existe una clara asimetría.

En el caso de la asimetría física, los servidores tienen potentes transmisores mientras que los clientes tienen receptores con limitada o nula capacidad de transmisión. Esto ocurre en la mayoría de las redes sin cables, por ejemplo, en las redes de satélite no existe canal de subida.

En algunos entornos, existen medios físicos separados con diferentes anchos de banda, mientras que otros utilizan un solo medio físico pero como denominador común el ancho de banda se reparte de forma desproporcionada entre transmisión y recepción, como por ejemplo en redes ADSL. En la siguiente tabla podemos ver algunas tecnologías y el ancho de banda dedicado en ambas direcciones.

Tabla 2.1. Ancho de banda de diferentes tecnologías de comunicación

Tipo de red	Ejemplo	Ancho de banda de bajada	Ancho de banda de subida
Satélite	DirecPC	400 Kbps	56 Kbps
CATV	Cable modem	10-30 Mbps	128 Kbps
Telefonía fija	ADSL	20 Mbps	1 Mbps
Wireless LAN	802.11b	10 Mbps	19 Kbps
UMTS Rel 5	Internet vía tarjeta SIM	3.6 Mbps	128 Kbps

En cuanto al volumen de información, las aplicaciones de acceso a información son asimétricas en cuanto que los datos intercambiados son mucho mayores en dirección fuente-destinatario. Típicamente un mensaje con una petición de información tendrá un tamaño muy pequeño mientras que la información solicitada será mucho mayor en tamaño. En estas aplicaciones los requisitos de ancho de banda de cada cliente son mucho mayores en bajada que en subida.

Por último, existen muchas aplicaciones en las que una gran cantidad de clientes solicitan un conjunto de información limitada de un pequeño número de servidores. Es muy frecuente que haya un subconjunto pequeño de datos muy demandado. En este entorno, si los servidores tienen que servir los datos individualmente, nos encontramos frente a dos problemas: por un lado se produce sobrecarga en los servidores, y por otro lado se produce sobrecarga en la red al enviar la misma información repetidas veces.

Los servidores acometen este problema simplemente limitando el número de conexiones aceptadas en un instante dado, lo cual resulta en interminables esperas en el acceso a sitios populares.

En un entorno de comunicación para dispositivos móviles (ver figura 2.4), el área geográfica de operación está dividida en regiones llamadas celdas, cada una de las cuales está bajo la influencia de un servidor o estación base. En principio podemos imaginar celdas de cualquier tamaño, desde varios kilómetros de diámetro hasta pocos metros.

Cada estación base estará conectada a la red fija para poder dar servicio a los clientes móviles que se encuentren dentro de la zona de cobertura de su celda. Los clientes móviles poseen movilidad, esto es, pueden moverse con libertad entre las diferentes celdas y por tanto pueden encontrarse bajo la influencia de diferentes estaciones base. Por el contrario, los clientes no móviles estarán directamente conectados a la red fija.

Existen variedad de aplicaciones tanto para celdas de área extensa (información de bolsa, noticias generales) como para celdas de área local (horarios de trenes/autobuses, información de tráfico, información en aeropuertos) e incluso para nanoceldas (espacios libres en parkings, estadios, restaurantes, publicidad en centros comerciales).

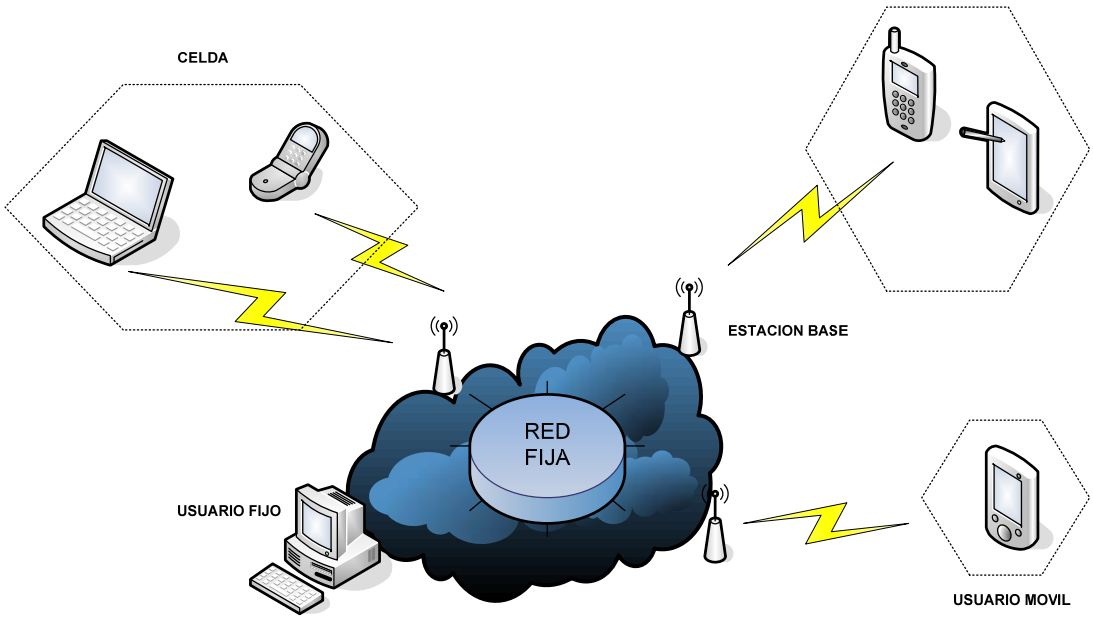

Figura 2.4. Entorno de comunicación para dispositivos móviles

Se asume que sólo existe un canal de difusión por celda. El servidor envía la información disponible en su base de datos de forma secuencial y cíclicamente por el canal a todos los clientes en su celda. Podemos pensar en un protocolo tipo TDMA (Time Division Multiple Access), como por ejemplo GPRS (General Packet Radio Service), mediante el cual el servidor divide el espacio temporal en ventanas, y en cada ventana temporal transmite un elemento de información (ver figura 2.5).

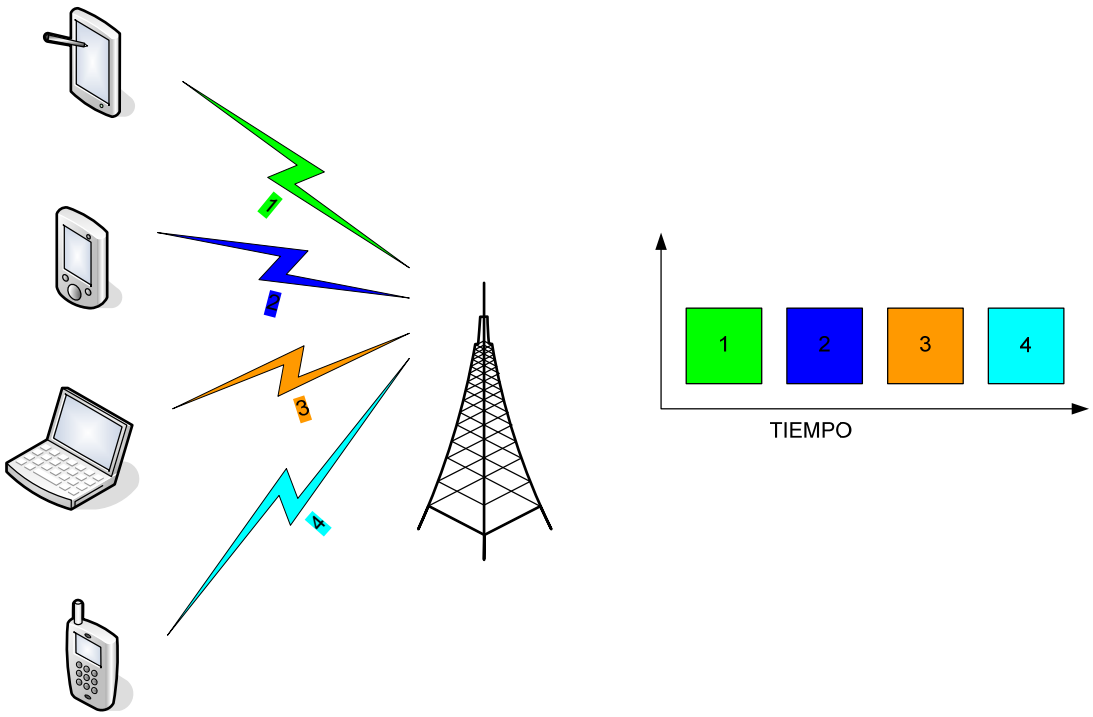

Figura 2.5. Protocolo tipo TDMA

Debido a las limitaciones de los entornos de comunicación para dispositivos móviles en cuanto a ancho de banda, batería, etc. las técnicas requeridas para el diseño de sistemas de comunicación eficientes al menor coste son diferentes a las de sistemas basados en redes fijas. Se requieren soluciones interoperables, escalables, fiables, eficientes y seguras. El factor determinante siempre es el ancho de banda disponible.

Con la proliferación de aplicaciones para redes inalámbricas y el aumento constante de usuarios de Internet, es muy necesaria la investigación para transmitir con eficacia la información. La investigación debe centrarse en algoritmos para el diseño de planificaciones de difusión, estrategias de *caching*, esquemas de indexado para reducir el consumo de energía. Las soluciones

eficientes en cuanto al consumo de energía son muy importantes, dado que las baterías en dispositivos móviles son un recurso muy limitado.

Debemos tener en cuenta que los entornos de comunicación para usuarios móviles son muy dinámicos en el sentido de que los clientes pueden conectarse/desconectarse del sistema en cualquier instante, pueden llegar/abandonar el área de influencia del servidor, los patrones de acceso a información suelen ser dinámicos (por ejemplo por la mañana los usuarios pueden demandar información de tráfico y por la noche información sobre eventos de ocio), y existe una gran varianza en el número de peticiones por unidad de tiempo.

2.4 Difusión: ¿Push o Pull?

Como ya hemos comentado, existen dos enfoques principales para transmisión de información: sistemas basados en push y en pull. Los sistemas push son aquellos en los que la comunicación es iniciada por el proveedor de información mientras que en los sistemas pull la comunicación es iniciada por el consumidor de información.

Ya hemos justificado el uso de la tecnología de difusión (una fuente, múltiples destinatarios) para aplicaciones de acceso a información en las que existe un número indeterminado de usuarios que poseen movilidad y por tanto perfiles dinámicos de acceso. En este contexto, la primera gran cuestión que se plantea es si la elección adecuada es un sistema basado en push o basado en pull.

En cualquier caso, consideramos que en cualquier sistema de comunicación que utiliza difusión existe una fuente y varios destinatarios conectados mediante una red de comunicación, en la cual existe al menos un canal de comunicación de gran capacidad en sentido fuente-destinatarios, llamado canal de bajada, y adicionalmente puede existir un canal en sentido contrario, llamado canal de subida.

Los sistemas de difusión basados en push [Acha95a, Acha95b] proporcionan alta escalabilidad diseminando la información a una gran cantidad de usuarios por medio de un canal de bajada (en sentido servidor-usuarios) de alta capacidad. El servidor trata de predecir las necesidades de los usuarios en base a perfiles o suscripciones preestablecidos, construyendo un programa que permite transmitir la información a toda la población de usuarios. El programa esencialmente determina el orden y la frecuencia con que los elementos de la base de datos son difundidos. El programa suele ser periódico, esto es, se transmite de forma regular y repetidamente a la comunidad de usuarios.

Los clientes en general son pasivos y la información les llega sin ser solicitada de forma explícita. Los usuarios monitorizan el canal de bajada y reciben los datos requeridos. Los elementos de información son auto-identificativos o bien se transmite un índice en el programa de difusión que permite a los usuarios identificar la información requerida. El sistema es de "sólo escucha" y no se requiere canal de subida. En estos sistemas la transferencia de información es siempre iniciada por el servidor (ver figura 2.6).

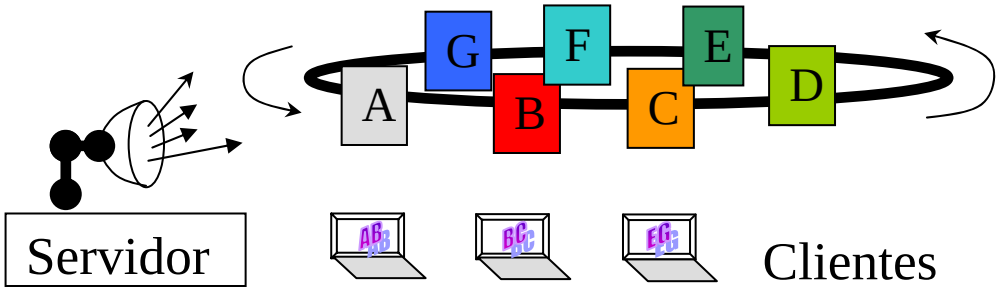

Figura 2.6. Sistema de difusión basado en push

Los sistemas push son análogos al sistema de televisión tradicional en el que los espectadores conocen la programación por medio de un listado de los programas que van a ser emitidos, pero no tienen control directo sobre el contenido de las emisiones. Los responsables de la programación la eligen basándose en índices de audiencia, es decir, en perfiles de acceso, dando prioridad a los programas más demandados.

La mayor ventaja del modelo push es su escalabilidad inherente. Las prestaciones del servidor no se ven afectadas por el número de clientes, con lo cual no existe un límite teórico de usuarios del sistema ya que el servidor no se satura. Es más, las prestaciones de cualquier cliente no se ven afectadas (al menos de forma directa) por la existencia de otros clientes monitorizando el canal de bajada. El ancho de banda disponible se utiliza de forma eficiente al satisfacer a muchos clientes mediante una sola transmisión.

El mayor inconveniente de los modelos push es la falta de interacción de los usuarios con el servidor, en este sentido los usuarios son siempre pasivos. Esto provoca la imposibilidad de adaptación a perfiles de usuario dinámicos (lo cual ocurre debido a la movilidad de los mismos) y la dificultad de asegurar el cumplimiento de restricciones temporales.

Cada elemento de información se transmite a todos los clientes, con lo cual si una fracción importante de la población no está interesada en algunos elementos, se malgasta el ancho de banda disponible. Además si existen muchos clientes con diferentes intereses, el número de elementos a transmitir para satisfacer esta demanda aumenta considerablemente.

Para que el modelo sea eficiente, el servidor necesita conocer con exactitud los perfiles de acceso a la información de los usuarios, para poder dar prioridad a los elementos más populares. Esta información de perfiles puede estar no disponible o no reflejar con exactitud las necesidades reales de los usuarios.

En el otro lado del espectro, encontramos el modelo de servidor basado en pull, en el cual el cliente envía una petición al servidor solicitando una información específica, el servidor localiza la información solicitada y responde a la petición enviando la información en cuestión. Recordamos que la información es enviada mediante difusión, esto es, todos los usuarios conectados a la red de comunicación tienen la posibilidad de recibirla aunque la petición haya partido de un usuario concreto.

Los modelos basados en pull incorporan un canal de subida (en sentido usuarios-servidor) de baja capacidad, que posibilita la transmisión de peticiones de los usuarios. El cliente siempre envía

de forma activa una petición al servidor para recibir la información requerida. El servidor encola las peticiones recibidas y las va sirviendo conforme a cierta política de planificación. En estos sistemas la transferencia de información es siempre iniciada por el cliente (ver figura 2.7).

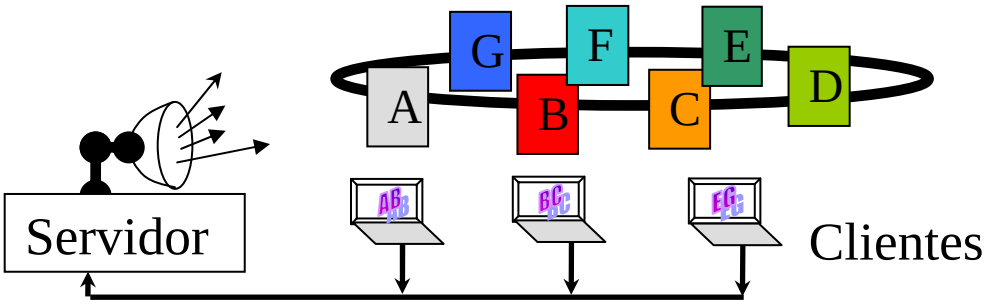

Figura 2.7. Sistema de difusión basado en pull

Los sistemas basados en pull tienen la ventaja de permitir a los usuarios solicitar la información que desean, en contraste con los modelos push en los que los usuarios sólo pueden monitorizar el canal en busca de información. Como contrapartida, necesitan un canal de subida que en algunos casos puede constituir un gasto importante o ser simplemente no factible debido a restricciones de infraestructura.

Los sistemas basados en pull son muy intuitivos y simples. Sin embargo, los avances en los últimos años en el mundo de la comunicación plantean dudas sobre la adecuación del modelo a un gran número de aplicaciones emergentes. El modelo pull funciona muy mal en condiciones de sobrecarga y además malgasta ancho de banda para aplicaciones con alta coincidencia de información.

La gran desventaja de los modelos pull es que sufren de falta de escalabilidad. A partir de un cierto número de peticiones por unidad de tiempo, el canal de subida se convierte en un cuello de botella del sistema y las peticiones empiezan a colisionar, causando la congestión del canal y degradando las prestaciones.

Además, debido a la limitación de recursos del servidor, existe un límite en el número de peticiones que pueden ser procesadas por unidad de tiempo. Esto se denomina tasa de servicio del servidor, y cuando la tasa de peticiones de los clientes supera a la tasa de servicio se produce la saturación del servidor. En los sistemas tradicionales la tasa máxima de peticiones puede ser estimada a priori, pero en aplicaciones emergentes con un enorme número de clientes esto no es trivial.

Aunque fuera posible incrementar los recursos, el diseño para el caso peor sería económicamente ineficiente ya que en el escenario del caso peor la carga del sistema suele ser muy superior a la del caso promedio.

En un intento de resolver los problemas encontrados en servidores push y pull, se han propuestos modelos *híbridos* [Acha97, Xuan97], que tratan de explotar la alta escalabilidad de los servidores push y la interacción con los usuarios de los servidores pull, estableciendo un compromiso entre los dos modos de funcionamiento.

En los modelos híbridos, los clientes utilizan el canal de subida para efectuar peticiones de elementos que no están planificados para transmitir por el servidor en un futuro próximo. Se suele

asumir que el canal de bajada y el de subida son independientes, es decir, el tráfico de uno no interfiere en el otro.

En los servidores híbridos (ver figura 2.8) existen dos modos de transmisión: el modo *difusión periódica*, en el cual la información es diseminada periódicamente, y el modo *difusión bajo demanda*, dedicado a la transmisión de información requerida explícitamente por los usuarios por medio del canal de subida. Se establecen dos canales lógicos de bajada, y se transmiten datos en modo periódico o bajo demanda mediante multiplexación por división temporal. Las prestaciones de este enfoque dependen en gran medida de la asignación del ancho de banda disponible a cada uno de los modos.

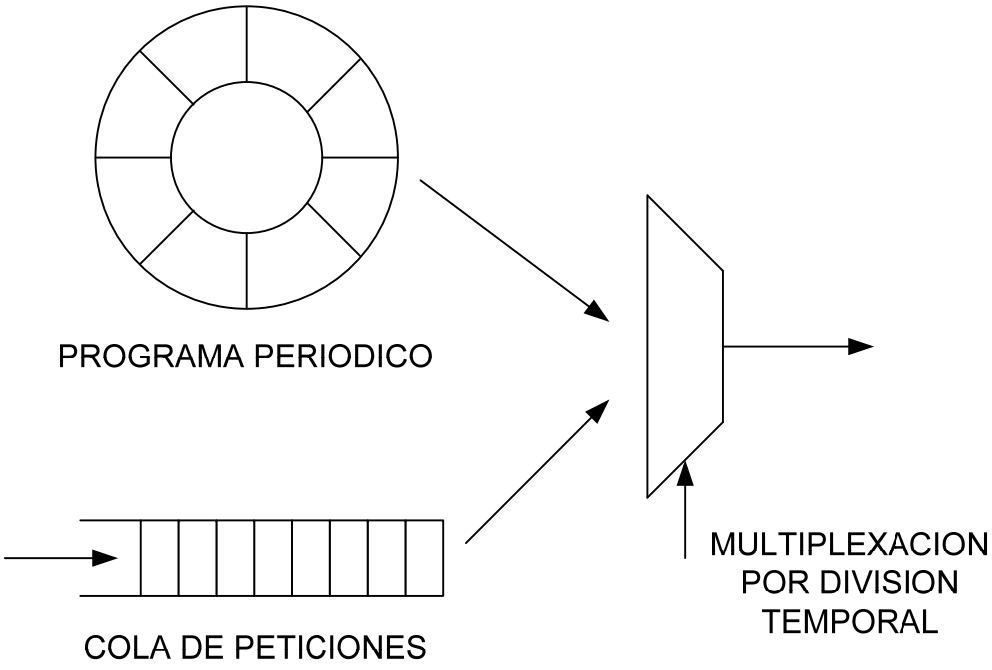

Figura 2.8. Sistema de difusión híbrido

En este tipo de sistemas deben existir mecanismos adecuados para el uso eficiente del canal de subida, ya que de lo contrario un gran número de peticiones saturaría el mismo, produciéndose multitud de colisiones y disminuyendo la eficacia del modelo.

2.5 Modelos basados en push

En los sistemas basados en push la información es diseminada de forma secuencial, en lugar de ser transmitida en respuesta a peticiones de clientes. Debido a ello, se necesita diseñar un programa de difusión donde los elementos de información aparezcan en una secuencia determinada con el objetivo de satisfacer a la mayoría de los usuarios con el menor tiempo de espera posible.

Esto se puede combinar con estrategias de caching en los clientes, de forma que localmente puedan memorizar una pequeña parte de la información previamente transmitida con la esperanza de que algún elemento de información a solicitar se encuentre en la caché y no haga falta esperar para recibirlo.

También se ha investigado profusamente sobre técnicas de indexado de información, de forma que los receptores, una vez recibido el índice, pasen a un modo de bajo consumo de energía (es decir, dejen de estar "a la escucha") y sólo se despierten en el momento preciso para recibir la información, con el consiguiente ahorro de energía.

2.5.1 Programa de difusión

Los primeros sistemas de difusión basados en push usaban una aproximación plana (por ejemplo Datacycle [Bowe92]), es decir, transmitían todos los elementos disponibles en la base de datos del servidor con la misma frecuencia. En esta aproximación todos los elementos de información son transmitidos secuencialmente del primero al último, tras lo cual se empieza otra vez por el primero repitiendo la misma secuencia, y así indefinidamente. El período usado para transmitir una vez la secuencia se llama ciclo de transmisión. Este algoritmo es muy simple, y cuando el servidor dispone de nuevos elementos de información los añade al final de cada ciclo.

El programa de difusión plano plantea el siguiente problema: en el caso habitual de que un gran número de clientes requiera el mismo elemento de información y exista un gran número de elementos de información en la base de datos, la probabilidad de que muchos clientes esperen un tiempo largo a recibir la información es alta. Dicho de otra forma, todos los clientes tienen el mismo tiempo de espera medio, es decir, la mitad de un ciclo de difusión, sin importar si reciben un elemento muy popular o muy poco popular.

La principal medida de prestaciones en este tipo de sistemas es el tiempo de espera promedio total (la media del tiempo de espera de cada usuario), con lo cual el diseño de un programa de difusión con el objetivo de minimizarlo es una cuestión de gran importancia.

En el trabajo de [Amma85], que consideraba únicamente elementos de longitud fija, se demostraba que para minimizar el tiempo de acceso medio a la información, el programa de difusión debía cumplir las siguientes 3 condiciones:

1. ser periódico

2. la varianza del interespaciado entre dos instancias consecutivas del mismo elemento debe ser minimizada

3. las frecuencias de transmisión deben ser proporcionales a la raíz cuadrada de las probabilidades de demanda

La condición 2 puede ser expresada de forma alternativa como sigue: las transmisiones de un elemento de información deben ser equiespaciadas siempre que sea posible. Si la tasa de peticiones para un elemento es fija, el tiempo de espera medio para cualquier petición que llegue en un instante aleatorio es la mitad del espacio de separación entre 2 transmisiones sucesivas del elemento.

Sin embargo, si las transmisiones no son equiespaciadas, la probabilidad de llegada de peticiones en un espacio mayor es más alta que en un espacio menor, con lo cual el tiempo de espera total medio aumentaría (habría más peticiones que esperan más frente a menos peticiones que esperan menos). Esta es la llamada "paradoja de la parada de autobús", por la cual el tiempo de espera medio aumenta a medida que aumenta la varianza del espacio entre transmisiones.

Asumiendo que las transmisiones de cada elemento de información son equiespaciadas, de la condición 3 se deriva la importante **regla de la raíz cuadrada** [Vaid99], para elementos de diferente longitud: "El tiempo de acceso para un elemento es minimizado cuando las instancias de este elemento son transmitidas de forma equiespaciada con una frecuencia proporcional a la raíz cuadrada de la demanda correspondiente al elemento, caracterizando la demanda mediante probabilidad de acceso, e inversamente proporcional a la raíz cuadrada de la longitud del elemento".

La regla de la raíz cuadrada puede ser enunciada formalmente como sigue: Dados M elementos con longitudes L_i y probabilidades de acceso P_i , el tiempo de respuesta total medio es minimizado cuando la frecuencia de difusión de cada elemento cumple la siguiente condición:

$$F_i \propto \sqrt{\frac{P_i}{L_i}}$$

En consecuencia, el espaciado de interdifusión S_i será directamente proporcional a la raíz cuadrada de la longitud del elemento e inversamente proporcional a la raíz cuadrada de la probabilidad de acceso del elemento:

$$S_i \propto \sqrt{\frac{L_i}{P_i}}$$

El modelo tradicional de difusión basado en push llamado *Broadcast Disks* (BD) [Acha95a, Acha95b] está basado en las 3 condiciones citadas anteriormente. En el modelo BD, el servidor de información repite periódicamente un programa planificado offline, basado en patrones de acceso a la información observados en el pasado. El programa periódico se planifica de acuerdo a las probabilidades de acceso, repitiendo con mayor frecuencia los elementos más demandados.

Un *ciclo de difusión* se define como la transmisión de una instancia del programa periódico. El programa periódico no varía con el tiempo y sólo existe un canal de bajada, de gran capacidad. El objetivo es minimizar el tiempo de acceso medio total, asumiendo que las instancias de un mismo elemento de información están siempre equiespaciadas.

Conceptualmente se puede considerar como un problema de asignación del ancho de banda disponible, con el objetivo de determinar el porcentaje de ancho de banda óptimo para cada elemento, dadas las probabilidades de acceso. Puesto que el ancho de banda total disponible es fijo, incrementar el ancho de banda para un elemento supone necesariamente disminuir el de otro(s).

El modelo asume que la población de clientes y los patrones de acceso asociados no cambian, y que los clientes no tienen capacidad de transmitir, esto es, no hay canal de subida, con lo cual el servidor no tiene ninguna información de los clientes. También se asume que la información es de sólo lectura, esto es, no existen actualizaciones de la información ni por parte del servidor ni por parte de los clientes.

El modelo BD puede ser interpretado como un sistema multidisco, en el cual los elementos menos demandados se ubican en discos que giran más lentamente que aquellos en los que se posicionan los elementos más demandados.

Podemos imaginar un disco como un subconjunto de elementos de la base de datos del servidor que contiene elementos con probabilidades de acceso similares. En cualquier caso, el espacio entre instancias consecutivas de un mismo elemento es el mismo siempre. En este modelo se asume que todos los datos tienen la misma longitud.

El servidor entrelaza las transmisiones de elementos de información de los diferentes discos, de forma que los elementos de los discos que giran más rápido (los más populares) se transmiten con mayor frecuencia. Todos los elementos de un mismo disco se transmiten con la misma frecuencia.

En la figura 2.9 se muestra un ejemplo con 5 elementos de información, distribuidos en 3 discos girando a frecuencias f1= 2f2 = 2f3. Podemos observar como el elemento I1 se transmite con el doble de frecuencia que el resto de elementos.

Asumiendo que tenemos M elementos en la base de datos y que utilizamos un número de discos K, a continuación enumeramos los pasos básicos para la construcción del programa periódico:

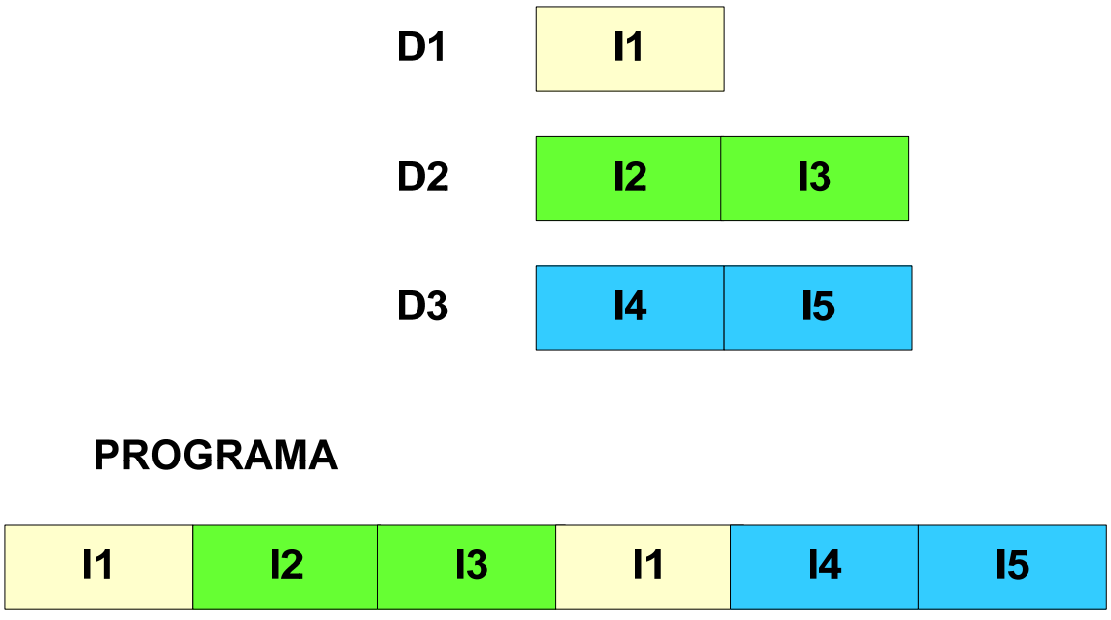

Figura 2.9. Ejemplo de modelo BD con 3 discos y 5 elementos de información

1. Ordenamos los elementos en orden descendente de probabilidad de demanda, esto es, de mayor popularidad a menor popularidad.
2. Asignamos los M elementos a los K discos de forma que cada disco contenga elementos de similar probabilidad de demanda.
3. Las frecuencias de giro de los discos fi deben ser múltiplo de la frecuencia de giro del disco más lento.
4. Dividimos los discos en sectores. El disco i se divide en max_sectores/fi, siendo max_sectores el mínimo común múltiplo de fi, con i entre 1 y K. Los sectores de diferentes discos pueden tener distinto tamaño, y en general pueden comprender más de un elemento de información.

5. El programa periódico se crea transmitiendo un sector del primer disco, después un sector del segundo, etc.

La figura 2.10 muestra el programa resultante para 11 elementos repartidos en 3 discos diferentes, con frecuencias de giro f1=2f2=4f3. Por lo tanto, el número de sectores de cada disco será 1, 2 y 4.

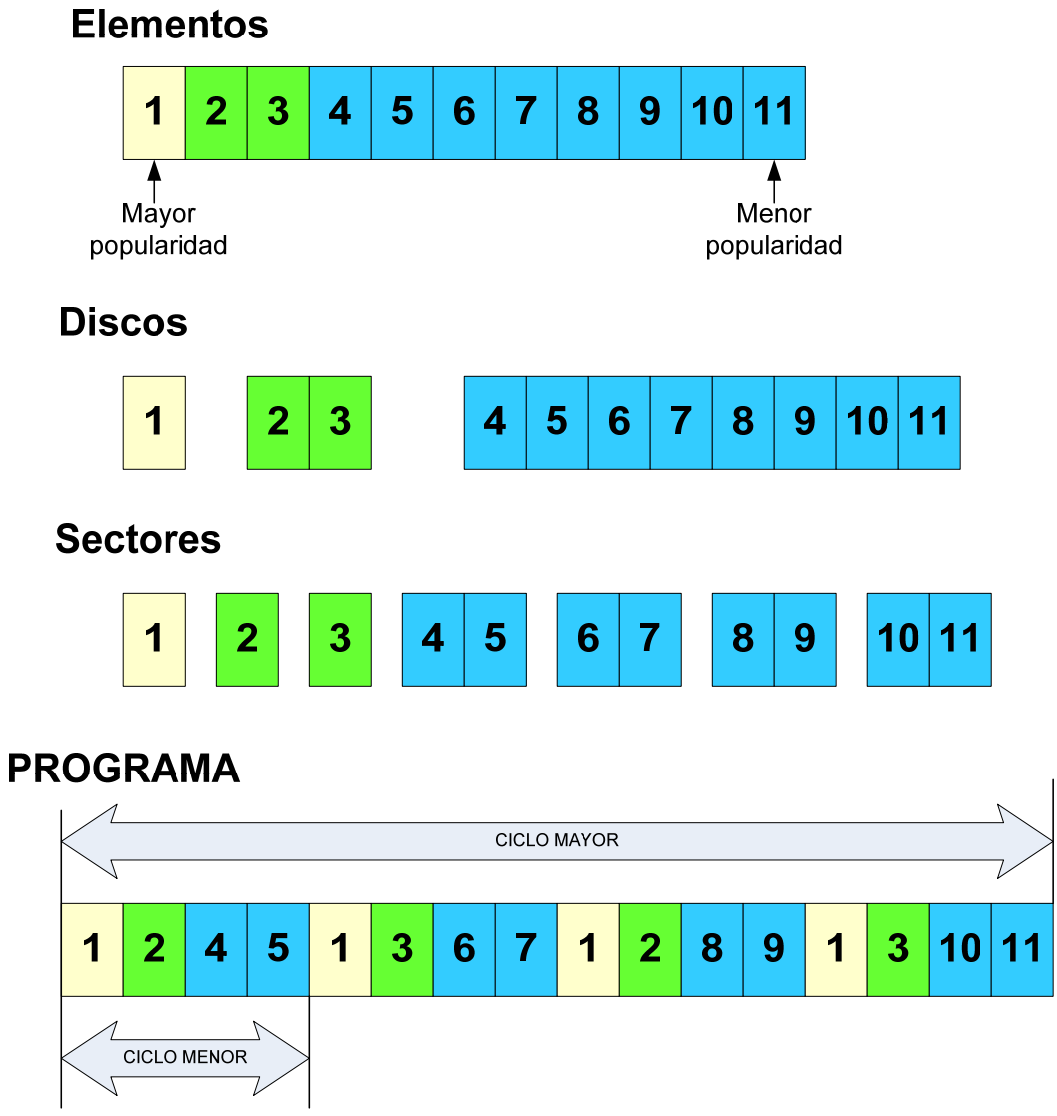

Figura 2.10. Ejemplo de modelo BD con 11 elementos de información

El programa resultante (llamado también ciclo mayor) contiene 4 ciclos menores, cada uno de los cuales contiene un sector de cada disco. El elemento con mayor popularidad es el de índice 1, transmitido con mayor frecuencia, mientras que el elemento menos popular es el de índice 11.

Nótese que el algoritmo provoca la aparición de espacios vacíos si no es posible la división de un disco en el número requerido de sectores. Por ejemplo, si en el ejemplo anterior M=10, quedaría un último espacio vacío (el correspondiente al antiguo dato 11) en cada ciclo de difusión.

El algoritmo especificado en BD no determina el modo de escoger los diferentes parámetros del mismo, es decir, el número de discos y las frecuencias relativas de giro de los mismos.

El método BD reduce el tiempo de espera medio de los datos más populares sacrificando la frecuencia de difusión de los datos menos populares. La limitación es que las frecuencias relativas deben ser estimadas con anterioridad. Las prestaciones mejoran con respecto a la aproximación plana, pero no se garantizan prestaciones óptimas. Además el modelo asume que todos los elementos tienen el mismo tamaño.

Las principales desventajas de este modelo son la falta de adaptación a las demandas de la población de usuarios (por ejemplo debido a la movilidad de los mismos), ya que no existe realimentación de los usuarios al servidor, la imposibilidad de garantizar el cumplimiento de plazos temporales y el gasto potencial de ancho de banda derivado del uso de un programa periódico estático.

Varios algoritmos de planificación han sido propuestos en el entorno del modelo BD, considerando factores tales como errores de comunicación, existencia de varios canales de bajada (*multichannel*), tamaños diferentes para los datos [Vaid99].

En principio podríamos pensar en utilizar la regla de la raíz cuadrada para generar una secuencia periódica óptima. Sin embargo, utilizando los valores obtenidos utilizando la regla para construir una secuencia periódica, algunos elementos se solaparían, es decir, tendrían que transmitirse en el mismo instante. Además, el costo computacional para construir un programa periódico utilizando la regla de la raíz cuadrada es demasiado alto para ser implementado en la práctica.

En [Vaid99] se propone un eficiente esquema (O(logM)) basado en "*packet fair scheduling*" (PFS), que se basa en conectar varias colas de entrada a una sola cola de salida. El objetivo es determinar el elemento que debe ser traspasado a la cola de salida en cada momento, tratando de distribuir de manera justa el ancho de banda disponible entre todas las colas de entrada.

El algoritmo primero calcula el espaciado de cada elemento basándose en probabilidades de acceso, y después transmite el elemento con el siguiente tiempo de difusión más próximo, definiendo el siguiente tiempo de difusión como la suma del último tiempo de difusión más el espaciado correspondiente al elemento.

Consideramos para cada elemento i los siguientes tres parámetros: *Bi*, *Si* y *Ci*. *Bi* es el instante más cercano en el que una instancia del elemento i podría ser transmitida, (se inicializa a 0 para todos los elementos), *Si* es el espaciado correspondiente al

elemento i, y *Ci* = *Bi* + *Si*. *Ci* puede ser interpretado como el instante en que la próxima transmisión de i debería realizarse en el caso peor.

El conjunto S contiene todos los elementos para los cuales *Bi* <= T, siendo T el instante actual. El algoritmo siempre elige para transmisión el elemento j con menor *Cj* de los del conjunto S. Después de transmitir el elemento j, actualiza *Bj* = *Cj*, *Cj* = *Bj* + *Sj*, y actualiza el conjunto S. Las simulaciones muestran que las prestaciones están muy cerca del algoritmo óptimo ideal.

A diferencia de BD, la difusión basada en PFS decide la secuencia de elementos a transmitir de forma dinámica, soporta datos con diferentes longitudes, y obtiene tiempo de respuesta medio total muy próximo al óptimo. No obstante, al igual que BD, está basada en probabilidades de acceso conocidas con anterioridad y en consecuencia no se adapta a patrones de acceso dinámicos.

La mayor parte de los trabajos se focaliza en el tiempo medio de respuesta como la única métrica de evaluación del algoritmo en cuestión. Sin embargo, la varianza del tiempo de respuesta puede considerarse como una medida de la calidad de servicio, ya que el tiempo de espera percibido por el cliente depende de su patrón de acceso. En [Jian98] se investiga cómo minimizar la varianza del tiempo de respuesta. Los resultados muestran un compromiso entre tiempo de acceso medio y varianza del tiempo de respuesta.

En [Jian99] se presta atención al tiempo de espera individual de cada cliente. En aplicaciones prácticas, después de un período de tiempo finito, al cliente ya no le es útil la información que ha solicitado. Este tipo de clientes "impacientes" obligan a tener en cuenta al algoritmo de planificación el ratio de servicio, definido

como la fracción de peticiones satisfechas antes de que el cliente abandone. Los resultados demuestran que existe un compromiso entre el tiempo de espera medio y el ratio de servicio.

El método propuesto en [Nico02] utiliza un autómata para proporcionar aprendizaje en el servidor, con el objetivo de conseguir adaptabilidad a las demandas de los usuarios, manteniendo baja la complejidad computacional.

Puesto que [Nico02] es un sistema basado en push, los clientes no tienen capacidad para efectuar peticiones. En su lugar, el sistema emplea realimentación simple por parte de los clientes. Los clientes emiten pulsos de energía a modo de confirmación en respuesta a cada elemento de información útil que reciben del servidor. El servidor capta la suma de todos los pulsos de energía y va adaptándose a la demanda real de la población, ya que el servidor recibe sumas de energía más altas en respuesta a la transmisión de los elementos más demandados.

2.5.2 Caching en el cliente

Caching se refiere a la capacidad de los clientes de almacenar elementos de información localmente una vez que han sido transmitidos por el servidor. De esta forma si el cliente requiere en el futuro la información almacenada localmente en la caché, el tiempo de acceso es mucho más rápido que si tuviera que ser enviada por el servidor. Además, se reduce el ancho de banda requerido ya que no es necesario efectuar peticiones al servidor si la información requerida está disponible localmente.

En [Acha95a, Acha95b], se demuestra que, para un modelo de difusión basado en push, las mejores prestaciones se obtienen cuando un cliente cachea aquellos elementos para los cuales la probabilidad de acceso local es mucho mayor que la frecuencia de difusión. Es decir, en estos sistemas, la regla intuitiva de hacer caching de los elementos de información que un cliente va a acceder con mayor probabilidad en el futuro no es válida [Barb94].

Al almacenar los elementos en una memoria de tamaño limitado, deben establecerse estrategias de reemplazo en caso de llenado de la misma. Una estrategia de reemplazo óptima es la que reemplaza el elemento residente con menor ratio entre probabilidad de acceso y frecuencia de difusión. El modelo se llama caché basado en coste [Acha95b].

Los mecanismos de gestión de caché son pasivos, esto es, sólo reaccionan cuando se solicita una página no residente en la caché. Una aproximación alternativa se denomina *prefetching* [Acha96], y consiste en que el dispositivo móvil almacena algunos elementos de información localmente a pesar de que no hayan sido solicitados con anterioridad.

En este tipo de sistemas la información fluye continuamente a los usuarios, por lo que el prefetching no tiene coste de recursos a nivel de red, únicamente un mayor consumo de espacio de almacenamiento y batería en los usuarios, a cambio de una potencial reducción en el tiempo de acceso a la información.

2.5.3 Indexado

Dado que el consumo de energía es importante, existen técnicas de indexado para evitar que los usuarios tengan que permanecer monitorizando el canal durante todo el tiempo de acceso a la información hasta que finalmente reciben la información requerida.

La idea básica es incluir información auxiliar en el programa de difusión acerca de los tiempos en que cada elemento será difundido según el programa. Así, los clientes son capaces de predecir en qué momento será difundido el elemento de información requerido, y permanecer en modo inactivo (mínimo consumo) para despertarse sólo en el momento justo.

El ancho de banda debido a esta información extra debe ser tenido en cuenta, y existen varias técnicas para multiplexar el índice con la información para mantener baja la latencia de acceso y ahorrar el máximo de energía.

El método más sencillo se llama *(1,m) indexing*, y consiste en transmitir el índice m veces en cada ciclo de difusión. En [Imie94a] se deriva el valor óptimo del parámetro m. También en [Imie94a] se habla de Distributed Indexing, mejora de la técnica anterior ya que se envía sólo la porción del índice que corresponde al segmento de datos que le sigue.

Otra de las opciones, *Index tree* [Imie97], es similar al árbol de índices que tenemos en un disco, sólo que en lugar de contener localización de registros del disco contiene tiempos de difusión de

los elementos de información. Para reducir el tiempo de acceso al índice, el índice puede ser replicado en su totalidad, o para evitar una sobrecarga excesiva, tener parte replicada (niveles superiores del árbol) y parte no replicada (niveles inferiores del árbol).

Por último, la técnica *Signature* [Lee96] es muy utilizada en sistemas de acceso a información. La signatura de una trama de información es un vector de bits generado haciendo *hashing* de los valores de la trama en cadenas de bits y superponiéndolas a continuación.

La signatura de cada trama es difundida al principio de la transmisión de la misma. Cada vez que un cliente desea acceder a cierta trama de información, genera internamente una signatura local de la trama. Después, el cliente va comparando la signatura local con las signaturas que va recibiendo (con una simple operación AND lógica de bits) y decide así si la trama que va a ser recibida a continuación contiene la información requerida. Cada vez que se recibe una signatura y no coincide con la local, el cliente puede pasar a estado inactivo y despertar de nuevo en el momento de la recepción de la siguiente signatura.

2.6 Modelos basados en pull

En cuanto a modelos de servidor basados en pull, siempre se considera la existencia de un canal de subida para recibir las peticiones de los usuarios. Tradicionalmente se han empleado algoritmos que sirven peticiones basándose en métodos heurísticos [Wong88] como:

- First Come First Served (FCFS) modificado, la cola tradicional se modifica de forma que si una petición para un determinado elemento se encuentra en la cola, nuevas peticiones son ignoradas porque serán satisfechas con una única transmisión.

- Most Request First (MRF), en el cual se sirve el elemento con mayor número de peticiones pendientes.

- Longest Wait First (LWF), se sirve el elemento con mayor tiempo total de espera (sumando los tiempos de espera de todas las peticiones para ese elemento).

En [Wong88] se concluye que con carga alta de peticiones el algoritmo LWF es el que da mejores prestaciones en cuanto a tiempo de espera total medio. Sin embargo este algoritmo tiene una complejidad que hace que su implementación no sea práctica, ya que se necesita calcular el tiempo de espera agregado para cada elemento de la base de datos.

El algoritmo R x W [Akso99] proporciona un método de prestaciones similares a LWF con una complejidad menor. R x W sirve los elementos teniendo en cuenta el número de peticiones y el tiempo de espera de las mismas. Concretamente sirve el elemento con máximo producto de número de peticiones R por tiempo de espera de la primera petición W. Así, el algoritmo da prioridad a los elementos más populares y a aquellos en los que algún usuario ha esperado un tiempo largo desde que envió una petición.

De esta forma, R x W combina los dos factores utilizados en los algoritmos MRF y FCFS, popularidad y tiempo de espera. Los

autores proponen una implementación con una versión parametrizada en la que hay un compromiso para minimizar el tiempo de espera medio y el tiempo de espera en el peor caso.

En un trabajo similar [Kara01] se presenta el algoritmo ATWT (Approximate Total Waiting Time), otra aproximación de LWF con una complejidad mucho menor, y a su vez menor también a la de R x W. ATWT minimiza el tiempo medio de espera y su varianza. Se propone una implementación del algoritmo utilizando *bucketing*, es decir, agrupando los elementos a servir en listas enlazadas y ordenándolas según el tiempo de llegada de la primera petición.

Cuando existen plazos temporales asociados a las peticiones, se han utilizado algoritmos como Earliest Deadline First (EDF, [Liu73]), en el cual se sirve primero el elemento cuyas peticiones asociadas tengan menor plazo relativo.

Recientemente ha aparecido el algoritmo SIN-α [Xu06], el cual se comporta mejor que EDF, MRF y RxW en entornos de comunicación asimétricos con restricciones temporales, para una amplia variedad de patrones de acceso. De hecho se demuestra mediante simulaciones que SIN- α se aproxima al teórico mínimo de plazos perdidos en condiciones de sobrecarga.

SIN-α integra los factores de urgencia de EDF y productividad de MRF, eligiendo para difusión el elemento con menor valor SIN-α en cada momento, siendo:

$$SIN-\alpha = \frac{TimeToFirstDeadline_i}{\left(PendingRequests_i\right)^{\alpha}}$$

Donde *TimeToFirstDeadline* es el espacio de tiempo entre el instante de decisión hasta que expire el plazo más próximo para el elemento *i*, *PendingRequests* es el número de peticiones no satisfechas, y α es un factor de ponderación de la productividad sobre la urgencia.

En el trabajo [Xu06], en ningún caso se tiene en cuenta el efecto de saturación del canal de subida ni las limitaciones en la potencia de computación del servidor.

2.7 Modelos híbridos

La primera arquitectura híbrida fue propuesta en [Wong88]. Los elementos de información eran clasificados como solicitados con mucha frecuencia o con escasa frecuencia. Los clientes debían saber a qué subconjunto pertenecía el elemento de información requerido. El modelo servía los elementos populares utilizando un programa periódico y los no populares bajo demanda. Utilizaba multiplexación por división en el tiempo para intercalar las transmisiones de ambos subconjuntos.

El modelo BD fue extendido en [Acha97], pasando a ser un modelo híbrido mediante la incorporación de un canal de subida, llamado IPP (Interleaved Push and Pull). El canal de subida se modela como una conexión punto a punto con el servidor. El servidor intercala elementos del programa periódico (calculado de la misma forma que en BD) con elementos enviados como respuesta a peticiones específicas por el canal de subida.

El ancho de banda es asignado a cada uno de los dos modos estáticamente, utilizando el parámetro *PullBW*, que puede variarse offline entre 0% (sistema push puro) y 100% (sistema pull puro). Si PullBW=50%, como máximo (ya que si no hay peticiones se transmite el siguiente elemento del programa periódico) se transmite un elemento de pull por cada uno transmitido del programa periódico. En este trabajo se demuestra mediante simulación que los modelos push y pull puros tienen mejores prestaciones en condiciones de sobreutilización e infrautilización extremas, mientras que el modelo híbrido proporciona buenos tiempos de acceso medio en todos los casos restantes.

Hay que destacar que un exceso de peticiones puede degradar las prestaciones del sistema. La cola de peticiones puede crecer mucho, incrementando los tiempos de espera. Además, si se llena la cola, algunas peticiones serán descartadas y no atendidas por el servidor.

Al igual que en BD, no hay adaptación a la población actual, ya que los contenidos del programa periódico no varían. Además, las peticiones no están asociadas a plazos temporales, y en ningún caso se tiene en cuenta la posible saturación física del canal de subida.

En un trabajo similar [Imie94b], se estudia el compromiso entre push y pull en un modelo híbrido. Proponen un algoritmo para dividir la base de datos en dos grupos, el grupo "broadcast" y el grupo "on-demand". El objetivo es mantener el tiempo de respuesta por debajo de un límite dado y minimizar el número de peticiones en el canal de subida, para lo cual se calcula el ancho de banda óptimo para cada modo.

Se transmiten periódicamente los elementos del grupo "broadcast", junto con un directorio del contenido para que los clientes puedan ahorrar energía, pasando a modo de bajo consumo tras recibir el directorio y despertándose en el instante preciso para obtener la información deseada. En [Imie94a, Imie97] se exploran métodos para transmitir este directorio o índice para minimizar tanto el tiempo de espera como el consumo de energía de los usuarios.

Al igual que en [Acha97], en [Imie94b] se asume que las frecuencias de acceso son conocidas a priori y en base a ellas se construye un programa periódico con los elementos del grupo "broadcast", transmitiendo con mayor frecuencia los elementos más demandados para minimizar el tiempo total de acceso. Al contrario que en [Acha97], sólo los elementos del grupo "on-demand" pueden ser solicitados en el canal de subida, y la cola de peticiones del servidor se considera infinita.

Tanto [Acha97] como [Imie94b] presentan el problema de no ser capaces de adaptar dinámicamente el porcentaje de ancho de banda destinado a cada modo (periódico o bajo demanda), frente a cambios en el grado de utilización del sistema.

En [Stat96, Stat97], se propone un modelo para la adaptación del ancho de banda dedicado a los modos periódico y bajo demanda. El servidor proporciona adaptación para minimizar la latencia promedio, incluyendo en el programa periódico los elementos demandados con mayor frecuencia.

El servidor cambia artificialmente la estimación de la probabilidad de demanda de los elementos, de forma que incluso los

elementos más populares son excluidos del programa periódico durante cortos periodos de tiempo. Esto causa un número de peticiones explícitas para estos elementos que son utilizadas por el servidor para obtener una estimación de la probabilidad de la demanda. De este modo, el modelo no necesita conocer las probabilidades de acceso. No se tiene en cuenta la capacidad limitada del canal de subida.

De manera análoga, [Datt99] presenta un algoritmo adaptativo que selecciona estadísticamente la información a ser difundida basándose en perfiles de acceso y usuarios registrados en cada celda. Los elementos son incluidos o excluidos del programa periódico basándose en su popularidad, la cantidad de tiempo que llevan sin servirse las peticiones asociadas, y el tiempo medio que un usuario permanece en la zona de influencia del servidor. Todas las peticiones son atendidas tarde o temprano, pero de nuevo las peticiones no se asocian con restricciones temporales.

En [Datt99] se consideran 2 tipos de estrategia: programa periódico de tamaño constante (CBS) y programa periódico de tamaño variable (VBS). En el primer caso, los elementos son incluidos en el programa periódico en orden de prioridad hasta completar el tamaño fijo asignado al programa periódico, mientras que en el segundo se incluyen en el programa periódico sólo aquellos elementos con valores altos de prioridad. Los resultados de simulaciones muestran que VBS es mejor cuando la carga del sistema es muy baja, mientras que CBS se comporta mejor en condiciones de sobrecarga.

En [Lee99] se propone un algoritmo que, de acuerdo con la carga actual del sistema, dinámicamente asigna el ancho de banda

disponible a los modos push y pull, y decide el modo de transmisión de cada uno de los elementos de información.

Otro sistema híbrido interesante es el propuesto en [Hu02]. Los autores proponen un modelo que utiliza la impaciencia de los clientes para desarrollar un algoritmo que estima en tiempo real los patrones de acceso a la información con la granularidad de un ciclo de difusión. Se basa en que los usuarios tienen una paciencia limitada esperando a que el elemento deseado se encuentre en el programa periódico. Después de esperar este tiempo, envían una petición para que sea servido bajo demanda. El servidor se programa para provocar ausencias deliberadas de los elementos del programa periódico, forzando a los clientes a enviar peticiones. Esto sirve al servidor para estimar las probabilidades de acceso.

En [Pino02], el servidor híbrido alterna la transmisión de los elementos más populares (push) con la de los menos populares (pull), utilizando un algoritmo de planificación basado en PFS para el subconjunto de elementos push y el algoritmo MRF para los elementos pull. El punto de corte entre elementos push y pull se elige de manera que el tiempo de acceso esperado del sistema sea minimizado. También se asume que el servidor conoce las probabilidades de acceso con antelación.

2.8 Modelos para peticiones con plazos

Los modelos descritos con anterioridad asumían que los clientes esperaban infinitamente a la llegada del elemento deseado por el canal de bajada. En un caso más realista, las peticiones de los clientes tienen plazos asociados, es decir, deben ser servidas en un cierto espacio de tiempo.

En entornos con restricciones temporales, minimizar el tiempo de espera total medio deja de ser el principal criterio a la hora de evaluar las prestaciones. El principal objetivo se convierte en maximizar el cumplimiento de plazos temporales.

Mecanismos para el cumplimiento de plazos temporales han sido integrados en el modelo BD (Real-Time BD [Baru97a, Baru97b, Best96]). Real-Time BD propone una organización de datos apropiada para garantizar el cumplimiento de plazos y un algoritmo, AIDA (Adaptive Information Dispersal Algorithm), que añade tolerancia a fallos mediante redundancia para recuperar información frente a errores de transmisión.

Real-Time BD propone tres alternativas para organizar los datos: plana, *rate monotonic* y *slotted rate monotonic*. La alternativa plana es la más simple pero malgasta el ancho de banda disponible. En rate monotonic, cada elemento se transmite con una frecuencia inversamente proporcional a su restricción temporal asociada, es decir, es como si cada elemento estuviera en un disco girando a diferente velocidad. En slotted rate monotonic, se combinan varios elementos en un mismo disco, de forma que estos elementos comparten la misma latencia en caso peor.

Sin embargo, el modelo Real-Time BD sigue siendo unidireccional, no hay canal de subida y el programa periódico sigue estando basado en una estimación previa de la distribución de las frecuencias de acceso.

El modelo Broadcast On-Demand (BoD), presentado en [Xuan97], introduce un servidor híbrido basándose también en una extensión de BD. BoD integra difusión periódica y bajo demanda,

considerando plazos temporales. BoD usa una fracción fija del ancho de banda disponible en el canal de bajada para el programa periódico, y el ancho de banda restante para el modo bajo demanda. BoD no es adaptativo a variaciones en la distribución de acceso, ya que el programa periódico es calculado offline basándose en distribuciones previas.

El servidor decide a priori los elementos que formarán parte del programa periódico (típicamente serán los de mayor frecuencia de acceso), y planifica un programa periódico en el que los elementos más populares son transmitidos más a menudo. La información transmitida periódicamente y la fracción de ancho de banda asignada a cada modo son constantes.

Debemos observar que en BoD, al ser un modelo basado en BD, la probabilidad de que se malgaste una cantidad importante de ancho de banda es alta, no sólo porque el programa periódico es constante, sino porque la frecuencia de difusión de un elemento no está relacionada con los plazos temporales asociados al mismo.

2.9 Conclusión

A modo de resumen, la tabla 2.2 lista las características de los principales modelos de servidores de información descritos en este capítulo.

En conclusión, ninguno de los modelos integra todas las características necesarias para funcionar de forma eficaz en un entorno de comunicación asimétrico con restricciones temporales en el que la distribución de acceso sea variable y el ancho de banda sea limitado tanto en el canal de bajada como en el de subida.

Tabla 2.2. Características de los principales modelos de servidores de información

Nombre	Tipo	Adaptativo	Plazos temporales	Ancho de Banda de subida limitado
Broadcast Disks	Push	No	No	Si
R x W	Pull	Si	No	No
SIN-α	Pull	Si	Si	No
Interleaved Push-Pull	Híbrido	No	No	Si
Real-Time BD	Push	No	Si	Si
Broadcast On Demand	Híbrido	No	Si	Si

En los siguientes capítulos se detallan las características necesarias que debería tener un servidor de información para entornos de comunicación asimétricos con restricciones temporales, y se presenta nuestro modelo de servidor AHB, el cual integra todas ellas con el objetivo de maximizar el índice de satisfacción de los usuarios en este tipo de entornos.

Capítulo 3:
Conceptos generales sobre servidores híbridos

En este capítulo introducimos la motivación principal para el empleo de servidores híbridos adaptativos en entornos de comunicación asimétricos con restricciones temporales y perfiles de acceso dinámicos.

En primer lugar enumeramos los requisitos que debe satisfacer un servidor de información en este tipo de entornos, para continuar detallando las principales características de los servidores híbridos. Seguidamente enunciaremos las cuestiones que deben ser resueltas cuando abordamos el diseño de los algoritmos de planificación necesarios en este tipo de servidores. Terminaremos mencionando los criterios empleados a la hora de medir cuantitativamente la eficiencia de estos sistemas.

3.1 Requisitos para servidores de información

Podemos considerar un entorno de comunicación asimétrico como un caso particular de un sistema de comunicación para dispositivos móviles, el cual es a su vez un caso particular de un sistema distribuido [Saty01].

Los sistemas de comunicación para dispositivos móviles imponen una serie de restricciones adicionales a los sistemas distribuidos, principalmente debidas a limitaciones más severas en el ancho de banda disponible y a limitaciones de recursos y de consumo energético en los dispositivos móviles.

A continuación enumeramos los requisitos que un servidor de información debe reunir para operar de forma eficiente en un entorno de comunicación asimétrico con restricciones temporales y perfiles de acceso dinámicos:

- **Uso eficiente del ancho de banda disponible**: Debido a las limitaciones inherentes a los entornos de comunicación asimétricos, el servidor debe utilizar el ancho de banda con el objetivo de satisfacer al máximo número de usuarios.

- **Alta escalabilidad**: La población de usuarios dentro de la zona de influencia del servidor es en teoría ilimitada, debido a la movilidad de los mismos, y en consecuencia necesitamos un modelo de servidor capaz de satisfacer a un número muy grande de usuarios sin disminuir las prestaciones.

- **Adaptación a perfiles de usuario dinámicos**: La población de usuarios móviles cambiará con el tiempo, lo que implica que el servidor debe ser capaz de adaptarse a diversos patrones de acceso.

- **Cumplimiento de requisitos temporales**: Existen aplicaciones en las cuales la información debe llegar al usuario antes de un cierto plazo para ser considerada útil. El servidor debe incorporar los mecanismos adecuados para tener en cuenta las restricciones temporales asociadas a las peticiones.

En general, los diferentes tipos de transmisión de información pueden ser categorizados en dos dimensiones [Xuan97], como se representa en la figura 3.1. La primera de las dimensiones se mueve en el rango que va de transmisión punto-a-punto a difusión. La segunda dimensión diferencia entre transmisión periódica y bajo demanda.

La interacción cliente/servidor puede ser considerada como la combinación de punto-a-punto y bajo demanda, mientras que el modelo Broadcast Disks sería difusión periódica. Otra posibilidad sería combinar difusión y bajo demanda, por ejemplo utilizando una política EDF con agrupamiento.

Sin embargo, la opción más apropiada para cumplir los requisitos mencionados anteriormente es utilizar un servidor de difusión híbrido (situado entre la opción periódica y la opción bajo demanda), ya que proporciona el potencial para una utilización eficiente del ancho de banda tanto del canal de bajada como de

subida, así como la capacidad de manejar requisitos temporales asociados a la información.

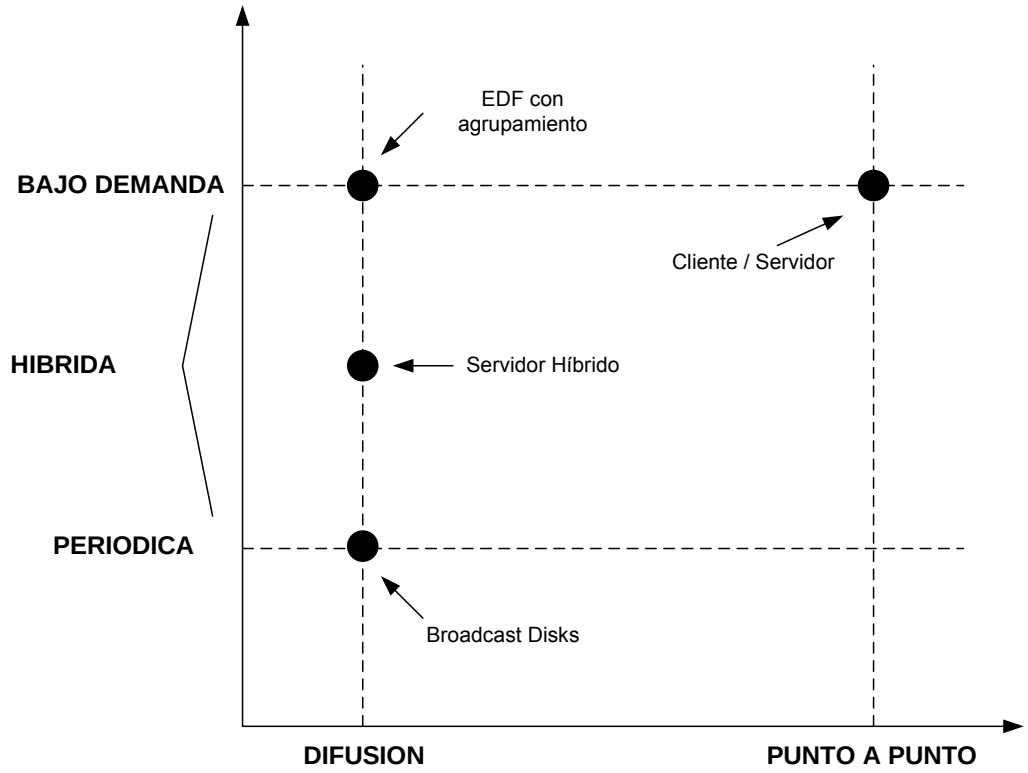

Figura 3.1. Diferentes políticas de transmisión de información

3.2 Características de los servidores híbridos

Como ya sabemos, un servidor híbrido es una combinación de un sistema push y un sistema pull, en el que algunos elementos son transmitidos por el servidor sin peticiones explícitas de los clientes mediante difusión periódica mientras que otros elementos son transmitidos sólo bajo solicitud, mediante difusión bajo demanda.

Asumimos que existe un único canal de comunicación físico en sentido de bajada y por lo tanto el ancho de banda disponible debe

repartirse mediante multiplexación por división temporal entre los dos modos de difusión existentes, periódico y bajo demanda.

El porcentaje de ancho de banda asignado a cada modo es una cuestión fundamental en el diseño de este tipo de servidores. En este sentido necesitamos que el servidor sea adaptativo, es decir, que este porcentaje pueda variar en función del tiempo para poder hacer frente a perfiles de acceso a la información dinámicos.

También es importante resaltar que debe existir una política de uso eficiente del canal de subida, ya que de lo contrario se produciría una saturación del mismo debido al gran número de clientes potenciales.

Como ya hemos mencionado, los servidores de información híbridos integran difusión periódica con difusión bajo demanda. Este modelo puede ser interpretado como dos servidores lógicos, el primero dándole más prioridad a la popularidad de la información (intuitivamente los elementos más populares deberían ser transmitidos periódicamente) y el segundo a los plazos temporales (tratando de cumplir los plazos de la información menos popular). La figura 3.2 muestra un sistema típico con un servidor de difusión híbrido.

Un parámetro importante en servidores híbridos es la longitud total del ciclo de difusión (Broadcast Length). Ésta se define para un determinado ciclo *i* como la suma de la duración del programa periódico (Periodic Broadcast Length) y la duración de la difusión bajo demanda (On Demand Length) durante dicho ciclo:

$$Broadcast\ Length(i) = Periodic\ Broadcast\ Length(i) + On\ Demand\ Length(i)$$

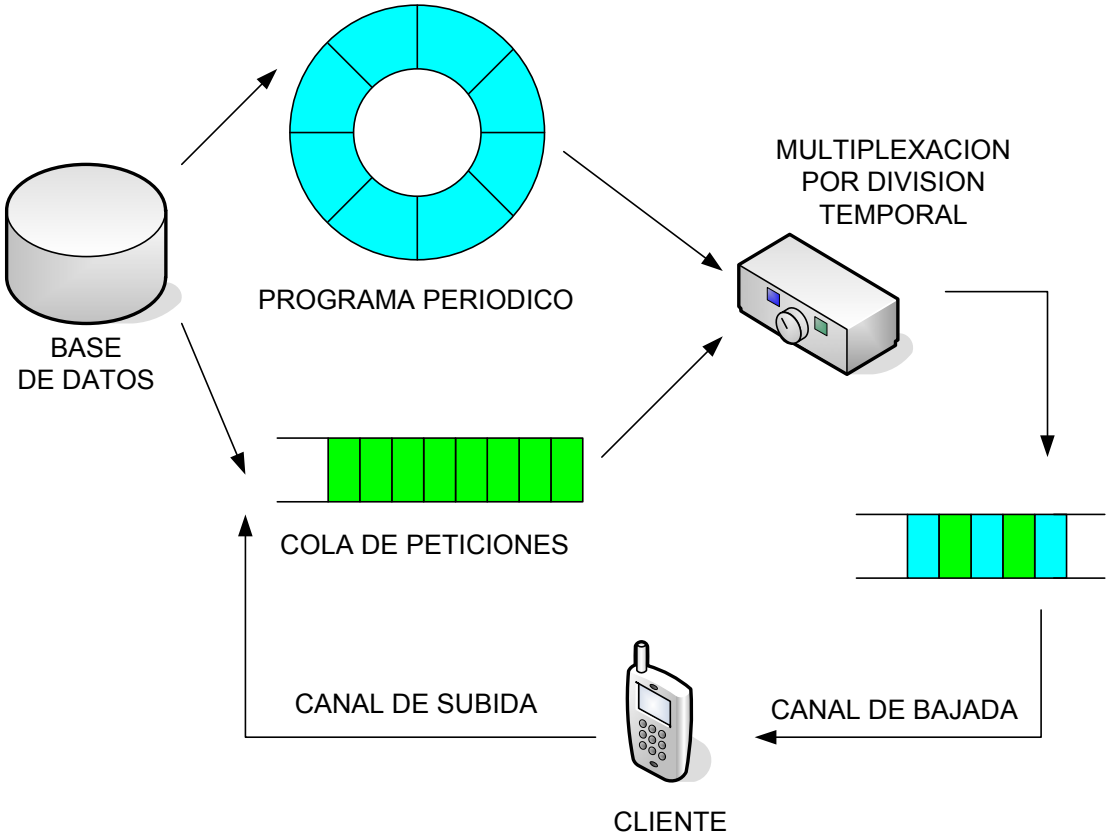

Figura 3.2. Esquema de servidor de difusión híbrido

La duración del programa periódico se refiere a la suma de los intervalos temporales donde se transmite la información escogida para ser difundida periódicamente. De forma similar, la duración de la difusión bajo demanda es la suma de los intervalos temporales donde el ancho de banda es asignado a la información escogida para ser difundida bajo demanda.

Resulta claro que la información incluida en el programa periódico no tiene que ser explícitamente solicitada por los usuarios. Asumiendo que un índice de la información transmitida es

intercalado con el programa periódico, los usuarios deben de consultar primero el índice para comprobar si la información que necesitan está siendo transmitida periódicamente y si la recibirían dentro del plazo temporal requerido antes de enviar una petición al servidor.

Este hecho disminuye en gran medida el número de peticiones que tienen que ser enviadas, evitando o reduciendo la saturación del canal de subida. Existen técnicas de indexado (por ejemplo, [Imie94a]), que ocupan sólo una mínima fracción del ancho de banda disponible en el canal de bajada manteniendo el tiempo de acceso al índice en niveles bajos. En los ejemplos mostrados en [Imie97], se manejan cifras entre el 2% y 4% de ocupación.

Es importante resaltar que la escalabilidad de un servidor de información de difusión está comprometida por la limitada capacidad del canal de subida, y por ello uno de los objetivos principales es intentar que el número de peticiones que deben ser enviadas por los usuarios sea lo menor posible.

Existen numerosos protocolos para la asignación del canal de subida a múltiples usuarios, la mayoría basados en CSMA (Carrier Sense Multiple Access) [Tann96]. En este tipo de protocolos de contención, sobrepasado un cierto flujo de datos, el canal se satura y el número de peticiones que no llega al servidor aumenta, debido a una alta probabilidad de colisiones. En consecuencia, la máxima capacidad del canal va decreciendo una vez que se sobrepasa el umbral de saturación, con lo cual la saturación del canal de subida puede ser modelada como un porcentaje de peticiones que no llegan al servidor.

Por citar un ejemplo, si la capacidad del canal de subida es de 30 Kbps y cada petición ocupa 150 bits, el canal de subida sólo puede acomodar 200 peticiones/seg. Para flujos superiores a 200, un determinado porcentaje de peticiones no llegarán al servidor, y este porcentaje aumentará según aumente el flujo de peticiones.

Adicionalmente, no debemos olvidar que cada petición que llegue al servidor debe ser planificada por el mismo, es decir, insertada en una cola de prioridad. El coste de inserción es O(logN), donde N es el número de peticiones en la cola. Este coste puede ser un factor importante dependiendo de la potencia de computación del servidor.

3.3 Cuestiones a resolver

A modo de resumen, las tres principales cuestiones en el diseño de los algoritmos de planificación necesarios para servidores híbridos adaptativos que operen en entornos de comunicación asimétricos con restricciones temporales y perfiles de acceso a la información dinámicos son las siguientes:

- **¿Cuál debe ser el porcentaje del ancho de banda del canal de bajada asignado a cada modo de difusión?** La cantidad de ancho de banda asignada a cada servidor lógico debería ser ajustada dinámicamente, dependiendo de la distribución de las frecuencias de acceso de los usuarios. Hay un compromiso entre la importancia relativa dada a la popularidad y a los plazos temporales.

- **¿Qué elementos deben ser difundidos periódicamente y cuáles bajo demanda, es decir, sólo cuando sean solicitados explícitamente?** En principio un elemento debe ser transmitido periódicamente cuando existe un ahorro potencial de ancho de banda con respecto a transmitirlo bajo demanda.

- **¿Cómo deben ser planificados el programa periódico y la transmisión bajo demanda?** Una vez decididos los elementos que deben ser difundidos periódicamente, debemos utilizar una política de planificación para decidir las frecuencias relativas de difusión y calcular el programa periódico de acuerdo con ellas. De forma similar, debe elaborarse la planificación de la difusión bajo demanda.

3.4 Diseminación eficiente de información en entornos de comunicación asimétricos

La eficiencia en el acceso a la información y el consumo de energía son dos cuestiones críticas en cualquier entorno de comunicación asimétrico. Los criterios que se han empleado con mayor frecuencia para medir las prestaciones de estos sistemas son:

- **Tiempo de acceso medio a la información**. El tiempo de acceso es el tiempo transcurrido desde el momento en que el usuario demanda la información hasta el momento en que la recibe. Además del tiempo de

acceso medio, existen otras métricas como el tiempo de espera total, la varianza de tiempo de espera, e incluso métricas que tienen en cuenta el tiempo de difusión de la información, como el tiempo de *stretch* [Acha98], definido como el ratio entre el tiempo de acceso y el tiempo de difusión.

Es obvio que la difusión de elementos poco demandados aumenta el tiempo de acceso y disminuye la eficiencia del sistema. Así, el programa de difusión, que determina qué elementos son difundidos y en qué momento, debe ser cuidadosamente diseñado.

- **Número de plazos perdidos.** En un entorno con restricciones temporales, el principal criterio a la hora de comparar la bondad de diferentes algoritmos de planificación no es el tiempo de acceso medio, sino que pasa a ser el número de solicitudes de información no satisfecho en plazo, es decir, el número de plazos perdidos. Este número suele darse en términos relativos, como un porcentaje del total de peticiones efectuadas.

- **Tiempo en modo activo**. El tiempo que los dispositivos móviles hayan de permanecer encendidos escuchando el canal de bajada para obtener la información es un indicativo del consumo de energía. El ahorro de energía es importante porque los usuarios móviles tienen baterías de duración limitada. La utilización de técnicas de indexado es necesaria siempre que tengamos restricciones de consumo en los clientes del servicio, lo

cual es muy frecuente en el caso de usuarios que posean movilidad.

- **Número máximo de clientes soportados por servidor**. En el caso de servidores híbridos, en principio no existe un límite en el número de usuarios, pero sí en el número de solicitudes que pueden ser procesadas por unidad de tiempo (no ocurre así en servidores push puros, que tienen en teoría escalabilidad infinita). En un servidor híbrido la limitación puede venir dada por el ancho de banda disponible en el canal de subida o por ocupación de recursos, principalmente capacidad de procesamiento del servidor.

Capítulo 4:

Análisis teórico del límite inferior de plazos temporales perdidos en servidores de difusión

En este capítulo analizamos el límite inferior del número de peticiones no satisfechas en plazo para un servidor de difusión en un entorno de comunicación asimétrico con restricciones temporales.

Tras efectuar una serie de consideraciones previas, detallamos el análisis matemático que nos llevará al cálculo del citado límite inferior.

4.1 Consideraciones previas

En el análisis que vamos a desarrollar a continuación asumimos lo siguiente:

- Tenemos un entorno de comunicación asimétrico en el cual existe un solo servidor de difusión y un número indefinido e ilimitado de usuarios

- El servidor tiene acceso a una base de datos donde se encuentran N elementos de información diferenciados

- Definimos una página de información como una unidad básica de transferencia de información entre el servidor y los usuarios, de longitud fija

- Por simplicidad, cada elemento de información tendrá una longitud igual a una página. El análisis es fácilmente extensible a elementos con diferentes longitudes

- Existe un único canal de bajada de alta capacidad, en sentido servidor-usuarios, mediante el cual el servidor transmite en modo difusión, es decir, cada elemento transmitido es visible a todos los potenciales usuarios

- Existe un canal de subida de baja capacidad, en sentido usuarios-servidor, mediante el cual los usuarios pueden solicitar elementos de información al servidor

- Cada mensaje de solicitud enviado por un usuario contendrá tres parámetros: el identificador del elemento de información solicitado, el instante en el que se ha producido la solicitud, y el plazo máximo para recibir la información

- Todos los elementos de información transmitidos por el servidor son auto-identificativos, es decir, llevan incluida una cabecera en la que consta su identificación

- Una vez efectuada una solicitud, el usuario monitoriza el canal de bajada hasta que aparece la información solicitada

- Toda la información está disponible en el servidor de forma local, no se realizan actualizaciones de la misma mientras el servidor está en funcionamiento

- No se utilizan técnicas de indexado

- Una petición se considera que está activa si su plazo no ha expirado aún. Una vez expirado el plazo máximo, la información carece de validez para el usuario

Podemos pensar en un sistema como el de la figura 4.1, en el cual tenemos un servidor con un canal de bajada de gran capacidad vía satélite, y una serie de clientes que efectúan peticiones a través de un canal de subida dedicado. El servidor recibe las peticiones en una cola de servicio y las atiende en base a una determinada política de planificación.

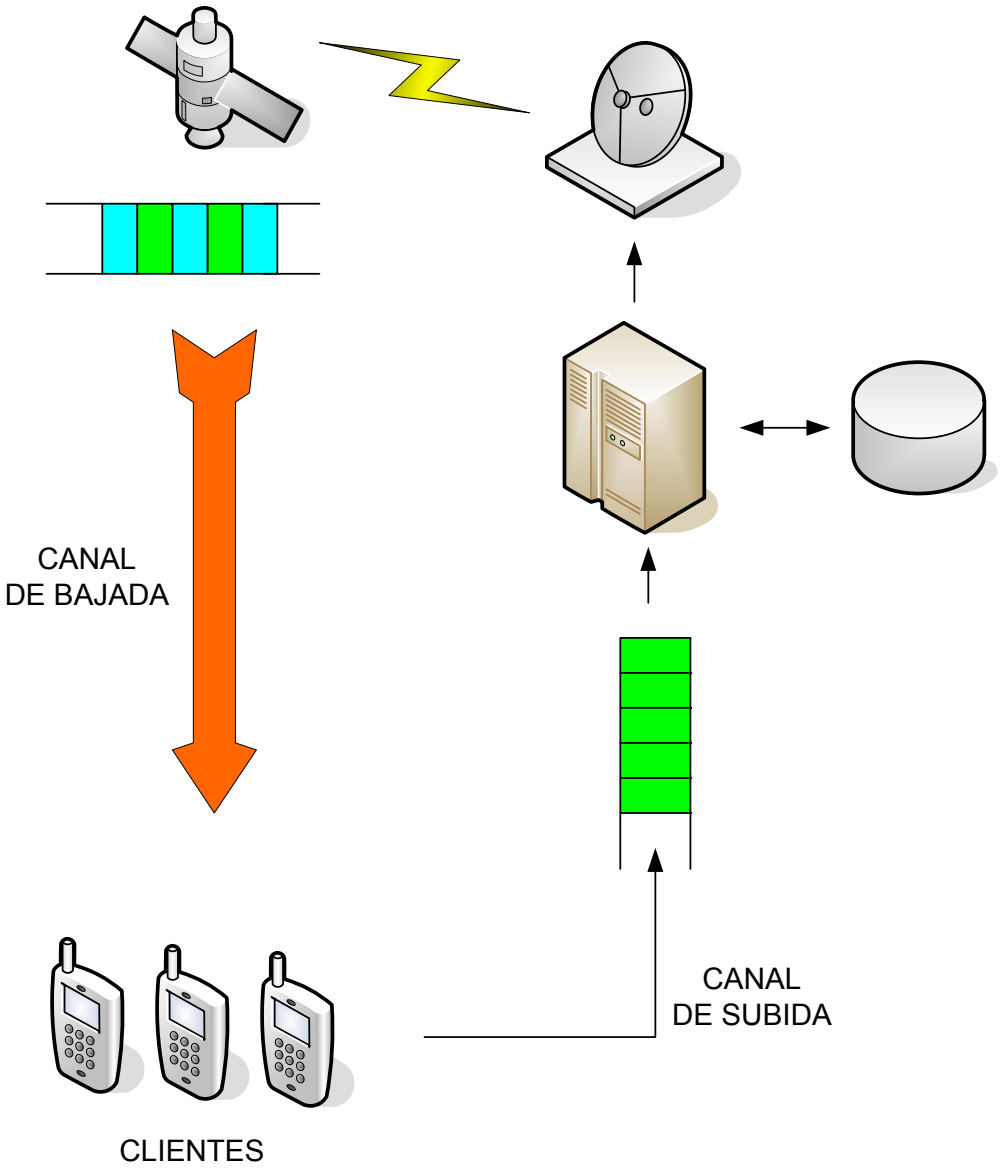

Figura 4.1. Modelo de servidor de difusión utilizado para el análisis

4.2 Detalle del análisis

Como ya hemos mencionado con anterioridad, en entornos de comunicación asimétricos con restricciones temporales, la eficacia de un servidor de difusión viene marcada por el ratio de peticiones no satisfechas, el cual se define como el número de peticiones que no son satisfechas en su correspondiente plazo dividido por el número total de peticiones efectuadas por los usuarios.

A continuación analizaremos el límite teórico inferior del ratio de peticiones no satisfechas cuando la tasa de peticiones es muy alta, es decir, en condiciones de fuerte sobrecarga para el servidor.

Los resultados del análisis pueden ser empleados tanto para comparación con diversos algoritmos de planificación como para aprovisionamiento de ancho de banda de difusión en sistemas.

En [Xu06] se analiza el límite teórico inferior del ratio de peticiones no satisfechas para tres distribuciones diferentes de plazos temporales: fija, uniforme y exponencial. Asumen que la distribución de plazos es la misma para todos los elementos de información, así como que todos los elementos de información poseen idéntica longitud, en concreto igual a una página de información.

A continuación obtendremos dicho límite para una distribución de plazos temporales más realista, aquella en la que conocemos el valor del plazo asociado a la solicitud de cada elemento de información. Por citar un ejemplo, podemos suponer que el servidor nos permite solicitar el elemento 28 de su base de datos con un

plazo de 2 minutos. Por simplicidad, al igual que en [Xu06], consideramos elementos de la misma longitud.

Nótese que en la distribución que analizaremos el plazo permitido en general puede ser diferente para cada elemento de la base de datos.

La notación empleada en el análisis se detalla en la siguiente tabla:

Tabla 4.1. Notación empleada en el análisis

Parámetro	**Notación**
Número de elementos	N
Tiempo entre transmisiones consecutivas del elemento i	b_i
Plazo relativo del elemento i	d_i
Tasa total de peticiones	λ
Tasa de peticiones del elemento i	λ_i
Probabilidad de acceso al elemento i	p_i

Si λ es la tasa total de peticiones, λ_i es la tasa de peticiones del elemento i y p_i es la probabilidad de acceso al elemento i, se cumple que $p_i = \lambda_i / \lambda$. Por ejemplo, si en total tenemos 200 peticiones por segundo, y para el elemento 7 tenemos 2 peticiones por segundo, la probabilidad de acceso al elemento 7 será de un 1% (0.01).

En una situación genérica, supongamos que dos transmisiones consecutivas del elemento i se producen en los instantes de tiempo t=0 y t=b (ver figura 4.2).

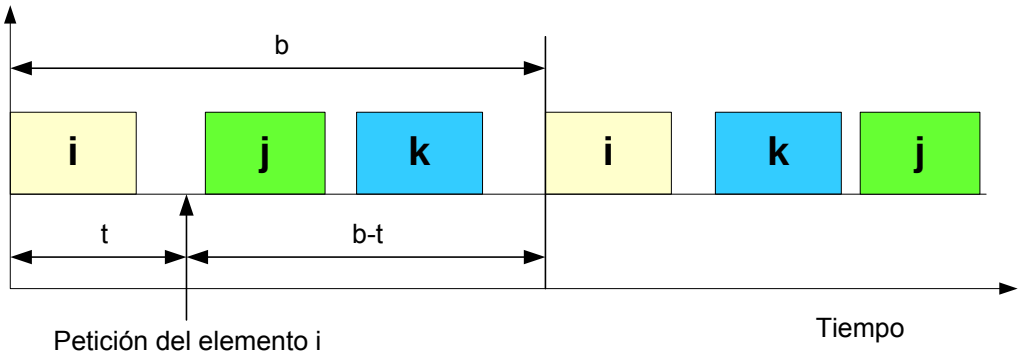

Figura 4.2. Situación genérica para 2 transmisiones consecutivas del elemento i

Ahora imaginemos que una petición del elemento i llega al servidor en un intervalo de tiempo infinitamente pequeño [t, t + Δt], tal que (0 < t ≤ b). Podemos observar que la petición no será satisfecha si y sólo si el plazo relativo solicitado es menor que (b−t).

Cuando la tasa de peticiones para el elemento i se aproxima a infinito, el número de peticiones para el elemento i que llegan en cualquier intervalo Δt infinitamente pequeño puede ser aproximada por $\lambda_i \Delta t$.

En consecuencia, tenemos que el ratio de peticiones no satisfechas para el elemento i en el intervalo [0, b] viene dado por la expresión:

$$\frac{\int_0^b F(b-t)\cdot\lambda_i\,dt}{\int_0^b \lambda_i\,dt} = \frac{1}{b}\int_0^b F(b-t)\,dt \qquad (1)$$

Siendo F(t) la función de distribución acumulativa de plazos temporales relativos, es decir, la expresión (1) nos da la probabilidad de que un plazo temporal relativo sea menor que t.

Como ya expusimos en el segundo capítulo, para minimizar el número de peticiones no satisfechas de un determinado elemento de información, las transmisiones deben ser periódicas y equiespaciadas siempre que sea posible, es decir, la varianza del interespaciado entre dos instancias consecutivas del mismo elemento debe ser minimizada [Amma85].

En lo sucesivo asumimos que los elementos de información son transmitidos de forma periódica, y llamaremos b_i (i = 1, 2,..., N) a la distancia entre dos transmisiones consecutivas del elemento i. El número de peticiones no satisfechas para el elemento i (lo llamaremos *DR*, del inglés *Drop Rate*) viene dado por la siguiente expresión:

$$DR(b_i) = \frac{1}{b_i}\int_0^{b_i} F(b_i - t)\,dt \qquad (2)$$

El número de peticiones no satisfechas en total (DR) será entonces:

$$DR\ (b_1, b_2, ..., b_N) = \sum_{i=1}^{N} \frac{\lambda_i}{\lambda} \cdot DR\ (b_i) = \sum_{i=1}^{N} \frac{p_i}{b_i} \int_0^{b_i} F\ (b_i - t)\, dt \qquad (3)$$

Si b_i es la distancia entre dos transmisiones consecutivas del elemento i, esto implica que la fracción 1/ b_i del ancho de banda disponible es utilizada para transmitir dicho elemento (recordemos que cada elemento tiene una longitud igual a una página). Por lo tanto, la siguiente condición debe cumplirse para utilizar el total del ancho de banda disponible en el canal de bajada:

$$\sum_{i=1}^{N} \frac{1}{b_i} = 1 \qquad (4)$$

A continuación obtendremos el mínimo DR total con la restricción (4) para una distribución de plazos temporales en la que el plazo asociado a cada elemento (d_i) es conocido a priori. La función de distribución acumulativa de plazos temporales relativos para esta distribución es la siguiente:

$$F\ (t) = \begin{cases} 0 & si \quad 0 \le t \le d_i \\ 1 & si \quad t > d_i \end{cases}$$

A continuación procedemos a calcular los valores de DR(b_i). La expresión (2) para nuestra F(t) puede evaluarse de forma gráfica, distinguiendo dos casos posibles:

1. Si b_i <= d_i, F(b_i-t) =0 y la expresión vale 0.
2. Si b_i > d_i, F(b_i-t) puede verse en la figura 4.3, y dado que la integral es el área bajo la curva, la integral entre 0 y b_i es

el área del rectángulo de lados 1 y b_i-d_i, es decir, b_i-d_i. Por tanto $DR(b_i) = (b_i-d_i) / b_i$.

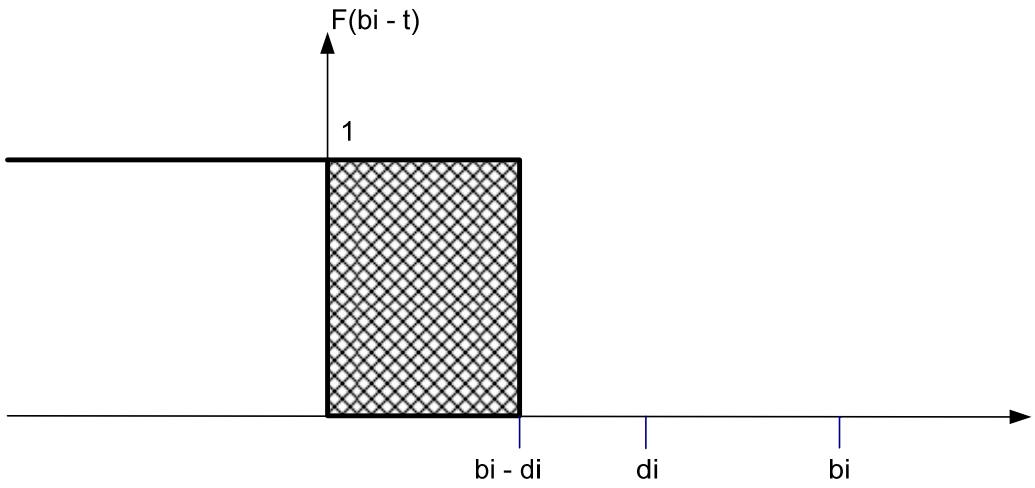

Figura 4.3. Valor de $DR(b_i)$ si $b_i > d_i$

En consecuencia:

$$DR\ (b_i) = \begin{cases} 0 & si \quad 0 \le b_i \le d_i \\ 1 - \dfrac{d_i}{b_i} & si \quad b_i > d_i \end{cases}$$

De forma gráfica, en la figura 4.4 vemos como el DR para un determinado elemento i es nulo siempre que el periodo de difusión no supere al plazo permitido para ser solicitado, creciendo a medida que aumentamos dicho periodo, tendiendo a 1 para valores muy elevados de b_i.

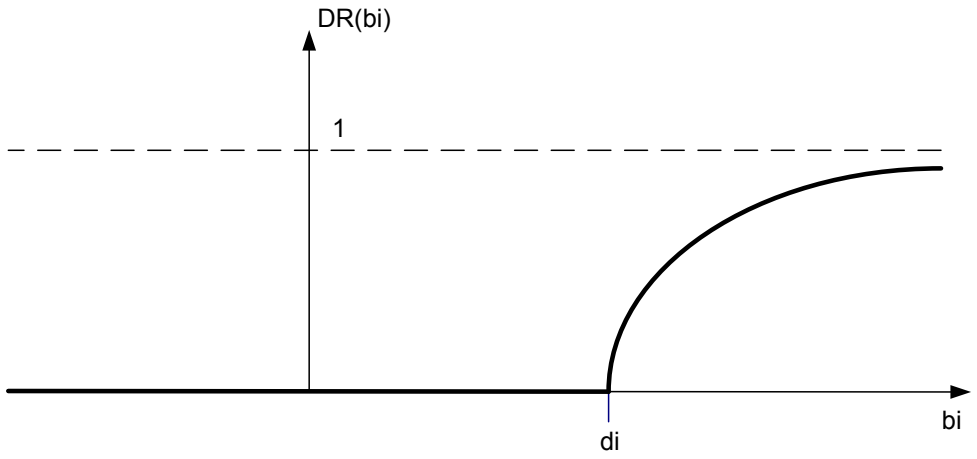

Figura 4.4. Representación gráfica de DR(b_i)

En resumen, necesitamos conocer el conjunto de distancias entre dos transmisiones consecutivas para cada elemento (b_1, b_2, ..., b_N) tal que DR total sea mínimo. Este problema de optimización se asemeja al problema conocido en términos algorítmicos clásicos como el problema "*0-1 knapsack*" [Corm90].

4.3 El problema "0-1 knapsack"

El enunciado del problema "*0-1 knapsack*" es el siguiente: un ladrón entra a una tienda donde se encuentran N objetos; el objeto i posee un valor v_i y pesa w_i kilos, donde v_i y w_i son números enteros. El objetivo del ladrón es llevarse el máximo valor posible, pero sólo puede cargar W kilos en su saco. Cada elemento puede ser tomado o dejado en su totalidad, pero no puede ser dividido en partes más pequeñas.

La primera aproximación que podríamos intentar para resolver el problema consiste en emplear la técnica simple de fuerza bruta. Enumeramos todos los posibles subconjuntos de elementos. Para cada subconjunto, comprobamos si cabe en el saco (es decir, si la suma de los pesos no supera el peso W), y si es así, comprobamos si su valor total (la suma de los valores v_i) es mayor que el de cualquier subconjunto comprobado con anterioridad, en cuyo caso es el subconjunto máximo provisional. Una vez que hayamos comprobado todos los subconjuntos existentes, tenemos la respuesta.

Desgraciadamente, la complejidad computacional del problema para N elementos es 2^N, con lo cual es conveniente encontrar otra manera de resolverlo de una forma más eficiente.

El problema "0-1 knapsack" posee la propiedad de "subestructura óptima". Esta propiedad dice que una solución óptima contiene dentro soluciones óptimas a los subproblemas contenidos en el problema.

Consideremos la carga de mayor valor cuyo peso es como máximo W kilos. Si quitamos un elemento cualquiera j de esta carga, la solución al subproblema restante debe ser la solución óptima para obtener la carga de mayor valor cuyo peso es como máximo W – w_j kilos incluyendo los N-1 elementos originales (excluyendo j).

Esta propiedad nos llevaría a pensar que el problema puede ser resuelto utilizando una estrategia "avara" (greedy). Esta estrategia consiste en llenar el saco eligiendo primero el elemento con el mayor "valor por kilo" (es decir, con mayor valor v_i/w_i), a

continuación el segundo mayor, etc., hasta que no haya espacio en el saco para ningún elemento más.

Por ejemplo, consideremos una tienda con tres elementos con pesos w_1=10, w_2=20 y w_3=30 y valores correspondientes v_1=60, v_2=100 y v_3=120, siendo la capacidad del saco W=50. Los valores por kilo serían entonces 6, 5 y 4 respectivamente. La estrategia avara nos lleva a una solución conteniendo los elementos 1 y 2, con un valor total de 160 (60 + 100). Es claro que la solución óptima es la que contiene los elementos 2 y 3, con un valor total de 220.

El problema "0-1 knapsack" puede ser resuelto utilizando programación dinámica "*dynamic programming*". En contraste con la estrategia avara en la que se toman las decisiones de una manera local, la programación dinámica compara de forma genérica diferentes candidatas a la solución óptima para determinar cuál es la más efectiva.

La efectividad de la programación dinámica se basa en almacenar la información apropiada para evitar el cálculo recurrente que se produce en subproblemas que se solapan, es decir, subproblemas que comparten sub-subproblemas.

Para resolver el problema "0-1 knapsack" mediante programación dinámica, primero debemos identificar los subproblemas que puedan ser resueltos de forma que sus soluciones puedan ser combinadas para resolver el problema total.

Definimos el subproblema S[k,w], como la solución óptima al problema 0-1 knapsack con k elementos y capacidad de saco w. Necesitamos encontrar una fórmula recursiva para S[k,w], en función de la solución que incluya cualquiera de los elementos 1 …

k-1. Si tenemos la solución óptima para un peso dado w incluyendo k-1 elementos, y consideramos el elemento k, tenemos dos casos principales:

1. w_k>w: en este caso, el peso del elemento k es mayor que el peso total, por tanto no puede ser incluido.

2. w_k<=w: en este caso, debemos calcular el valor que tendríamos si incluyéramos el elemento k. Este valor será igual al del elemento k más el valor máximo obtenido con los k-1 elementos previos, con un peso total de w – w_k. En otras palabras, S[k-1,w-w_k] + v_k. Si este valor es mayor que el máximo para los primeros k-1 elementos, el elemento k será incluido.

Por lo tanto si el elemento k es finalmente incluido el nuevo valor total será S[k-1,w-w_k] + v_k, y si no es incluido el valor no cambiará, es decir, seguirá siendo S[k-1,w].

En consecuencia, la definición recursiva para la solución del problema es la siguiente:

$$S[k,w] = \begin{cases} S[k-1,w] & si \quad w_k > w \\ Max\{S[k-1,w], S[k-1,w-w_k] + v_k\} & si \quad w_k \le w \end{cases}$$

Para obtener una implementación eficiente, debemos guardar los resultados intermedios de los subproblemas. Podemos ver los subproblemas como si ocuparan un espacio bidimensional, siendo un eje los elementos que podemos elegir (de 1 a N) y el otro eje el peso que llevamos (de 0 a W). Para un elemento cualquiera k, podemos encontrar todos los valores de S[k,w] desde w=0 a w=W.

El pseudocódigo para la construcción de la tabla sería el siguiente:

```
KNAPSACK(N,W)
{

      // inicializamos
      for (w=0; w<W; w++)
            S[0,w] = 0;

      for (i=0; i<=N; i++)
            S[i,0] = 0;

      // recorremos las 2 dimensiones
      for (i=1; i<=N, i++) {
            for (w=0; w<W; w++) {
                  if (w(i) <= w) {
                        if ( S[i-1,w-w(i)]+v(i) > S[i-1,w] ) {
                              S[i,w] = S[i-1,w-w(i)]+v(i);
                        }
                        else {
                              S[i,w] = S[i-1,w];
                        }
                  }
                  else {
                        S[i,w] = S[i-1,w];
                  }
            }
      }
}
```

Si tenemos, por ejemplo, un conjunto de elementos con (w_i, v_i)={(2, 3), (3, 4), (4, 5), (5, 6)}, con una capacidad de saco W=5, el resultado obtenido por el algoritmo sería el siguiente:

	w = 0	w = 1	w = 2	w = 3	w = 4	w = 5
i = 0	0	0	0	0	0	0
i = 1	0	0	3	3	3	3
i = 2	0	0	3	4	4	7
i = 3	0	0	3	4	5	7
i = 4	0	0	3	4	5	7

Esencialmente la tabla nos dice el valor máximo que podemos meter en el saco para cualquier peso del saco w entre 0 y W, utilizando todos los elementos hasta i, con i entre 0 y 4. La solución al problema total es S[N,W]=7.

Para encontrar los elementos que deben ser incluidos para conseguir el valor anterior, debemos observar que si S[i,k] es distinto de S[i-1, k] es porque el elemento i ha sido incluido, ya que hemos añadido valor al saco. Por lo tanto el algoritmo que nos da los elementos en cuestión es:

```
FIND-KNAPSACK-ITEMS(N,W)
{
      w=W;

      for (i=N; i>=0, i--) {
            if ( S[i,w] != S[i-1,w] ) {
                  // el elemento i es incluido
                  w = w - w(i);
            }
      }
}
```

En el ejemplo anterior, obtenemos que los elementos a incluir son el 1 y el 2, con un peso total de 5 kilos y un valor total de 7.

La confección de la tabla tiene una complejidad O(n x W), mientras que el algoritmo para encontrar los elementos puede ser efectuado en O(n), lo cual es una mejora significativa a la técnica de fuerza bruta, que recordamos tenía una complejidad de $O(2^N)$.

4.4 Aplicación del problema knapsack a nuestro análisis

En nuestro caso, podemos asumir que $b_i=d_i$ para cualquier elemento que sea transmitido periódicamente, y en consecuencia $DR(b_i) = 0$ para todos los elementos elegidos, hasta completar el ancho de banda disponible total. En este caso nuestro problema es equivalente al 0-1 knapsack.

El peso de cada elemento es $1/b_i=1/d_i$, y el valor de cada elemento es p_i. Por lo tanto utilizando programación dinámica podemos encontrar los elementos que hacen que el DR total sea minimizado, con la restricción de no superar el ancho de banda total disponible.

De forma intuitiva, el DR mínimo se alcanza asignando un período de transmisión igual a d_i para los elementos capaces de satisfacer más peticiones por unidad de ancho de banda, sujeto a la restricción del ancho de banda total disponible.

Ahora bien, es necesario contemplar la posibilidad de considerar valores posibles para b_i mayores a d_i. En este caso nuestro problema, además de exhibir la propiedad de "subestructura óptima", también tiene la propiedad de elección avara "*greedy choice*". Esta propiedad sustenta que una serie de

elecciones óptimas de forma local pueden ser combinadas para componer una solución óptima a nivel global.

Así, en cualquier paso en la ejecución de un algoritmo que resuelva un problema con estas dos propiedades podemos efectuar la elección mejor de acuerdo con la función de valoración correspondiente y podemos estar seguros de que nos llevará a la solución óptima. Es decir, una vez que se realiza una elección, nunca se vuelve atrás.

Nuestro problema se asemeja más ahora al problema "fractional knapsack", en el cual el ladrón puede tomar porciones de elementos. El citado ejemplo del 0-1 knapsack con pesos w_1=10, w_2=20 y w_3=30 y valores correspondientes v_1=60, v_2=100 y v_3=120, siendo la capacidad del saco W=50, puede resolverse entonces con la estrategia avara, tomando primero el elemento 1 (llenamos 10, valor 60), después el 2 (llenamos otros 20, valor 100) y después la fracción del elemento 3 necesaria para llenar el saco, es decir 2/3 (llenamos los últimos 20, valor 120 x 2/3 = 80), para completar un valor total de 240, siendo la solución óptima.

En nuestro problema, los elementos que satisfacen más peticiones por unidad de ancho de banda son aquellos que poseen los mayores valores del producto: $r_i = p_i \times d_i$.

Si tenemos solamente dos elementos en nuestro sistema, que podemos llamar 1 y 2 sin pérdida de generalidad, y se cumple que $r_1 > r_2$, si disminuimos el ancho de banda asignado a 1 y lo incrementamos a 2 en la misma cantidad, se produce un incremento de DR total positivo. A continuación se detalla la demostración. Tenemos que:

$$DR\ total = (p_1 - p_1 \times d_1/b_1) + (p_2 - p_2 \times d_2/b_2)$$

Llamemos $k_i = 1/b_i$ al ancho de banda asignado a i. La expresión queda como sigue:

$$DR\ total = (p_1 - k_1 \times p_1 \times d_1) + (p_2 - k_2 \times p_2 \times d_2)$$

Sabemos que se cumple $k_1 + k_2 = 1$, por la restricción (4). Si aumentamos el ancho de banda asignado al elemento 2 en Δk y disminuimos el ancho de banda asignado al elemento 1 en la misma cantidad, tenemos que el incremento producido en DR es:

$$\Delta DR\ total = \Delta k \times (p_1 \times d_1) - \Delta k \times (p_2 \times d_2) =$$
$$= \Delta k \times (p_1 \times d_1 - p_2 \times d_2) > 0$$

En consecuencia siempre que haya una cantidad de ancho de banda no asignada, debemos asignarla al elemento disponible con mayor valor del producto $p_i \times d_i$, es decir, nuestro problema posee la propiedad "greedy choice", lo cual nos permite resolverlo mediante la estrategia avara.

Sea S el conjunto de elementos con los mayores valores de r_i y que cumplen la restricción (4). Sea j el elemento con mayor valor del producto $p_j \times d_j$ de todos los que no hayan sido incluidos en S debido a la restricción (4).

Utilizando la estrategia avara, estamos en disposición de afirmar que los periodos de transmisión con los que se alcanza el DR mínimo son los siguientes:

$$b_i = \begin{cases} d_i & i \in S \\ \dfrac{1}{1 - \sum_{i \in S} \dfrac{1}{d_i}} & i = j \\ \infty & i \notin S, i \neq j \end{cases}$$

Es decir, transmitimos los elementos que satisfacen más peticiones por unidad de ancho de banda con un período igual al plazo mínimo asociado, hasta completar el ancho de banda disponible. Si existe ancho de banda remanente, lo utilizamos para transmitir el elemento que satisface más peticiones por unidad de ancho de banda de los que no hemos transmitido aún.

Teniendo en cuenta que utilizando la estrategia avara el DR será cero para todos los elementos del conjunto S, el DR mínimo puede calcularse mediante la expresión:

$$DR\ (b_1, b_2, ..., b_N) = \sum_{i \notin S} p_i + p_j \cdot \left(1 - \frac{d_j}{\dfrac{1}{1 - \sum_{i \in S} \dfrac{1}{d_i}}} \right) \qquad (5)$$

En resumen, el procedimiento para calcular el DR mínimo para una distribución de plazos temporales en la que se conoce el plazo asociado a las solicitudes de cada elemento es el siguiente:

1. Construir el conjunto S tomando de uno en uno los elementos con los mayores valores del producto $r_i = p_i \times d_i$, sin superar el ancho de banda disponible, esto es, cumpliendo la restricción (4)

2. Tomar el elemento con mayor valor del producto ri que no esté incluido en S (lo hemos llamado elemento j), y asignarle el ancho de banda restante

3. Utilizar la expresión (5) para calcular el DR mínimo, mediante la suma de las probabilidades de acceso de los elementos no incluidos en el conjunto S más el DR correspondiente al elemento j

Si tenemos, por ejemplo, un conjunto de elementos caracterizados por (p_i, d_i):

Elemento	**p_i**	**d_i**	**r_i**	**$1 / d_i$**
1	0.1	4	0.4	0.25
2	0.2	5	1	0.2
3	0.05	6	0.3	0.17
4	0.15	8	1.2	0.13
5	0.2	10	2	0.1
6	0.15	5	0.75	0.2
7	0.1	11	1.1	0.09
8	0.05	12	0.6	0.08

El resultado obtenido por el algoritmo para el cálculo del DR mínimo sería el siguiente:

- Conjunto S: elementos 5,4,7,2,6,8 (ancho de banda total = 0.1 + 0.13 + 0.09 + 0.2 + 0.2 + 0.08 = 0.8)
- Elemento j: elemento 1 (ancho de banda 20% restante)
- DR = $p_3 + p_1 \times (1 - d_1/(1-1/d_5-1/d_4-1/d_7-1/d_2-1/d_6-1/d_8))$ = 0.05 + 0.1 x (1 – 4 / 1 / 0.2) = 0.05 + 0.1 x (1/5) = 0.07

El cálculo del DR mínimo teórico nos será de gran utilidad a la hora de evaluar comparativamente las prestaciones de modelos reales de servidores de difusión, y en particular del modelo de difusión híbrido adaptativo que presentaremos en el siguiente capítulo.

Capítulo 5:

Modelo de servidor de información híbrido adaptativo AHB

En este capítulo proponemos nuestro modelo de servidor de difusión híbrido adaptativo, al que denominaremos en lo sucesivo AHB (Adaptive Hybrid Broadcast).

Detallamos los algoritmos de planificación empleados por AHB tanto para la selección de la información servida de forma periódica como para la difusión de información bajo demanda. También se expone la técnica empleada para la estimación de las frecuencias de acceso a la información por parte de los usuarios.

5.1 Introducción

Como ya se ha expuesto en anteriores capítulos, la noción de servidor híbrido se apoya en categorizar los elementos de información en dos grupos diferenciados, los elementos servidos de forma periódica y los elementos servidos bajo demanda.

Nuestro servidor híbrido AHB elabora un programa con los elementos pertenecientes al primer grupo y lo transmite a intervalos regulares en el tiempo, mientras que recibe simultáneamente peticiones de los elementos menos demandados y las va atendiendo de acuerdo a un determinado criterio de selección. En este sentido se diferencian los dos posibles modos de transmisión: periódico y bajo demanda.

El objetivo de nuestro modelo de servidor híbrido AHB es adaptar la información servida periódicamente y la cantidad de ancho de banda destinada a cada uno de los dos modos de transmisión (periódico y bajo demanda) para utilizar el ancho de banda global disponible de la forma más eficiente posible y así minimizar el número de plazos perdidos.

En principio el modo periódico debe utilizarse para elementos útiles a una gran cantidad de receptores, mientras que el modo bajo demanda debe utilizarse para elementos con baja popularidad.

El programa periódico es calculado dinámicamente al final de cada ciclo de difusión, y se incluyen en el mismo sólo los elementos que potencialmente producirán un ahorro de ancho de banda en el canal de bajada al transmitirlos periódicamente.

Los elementos incluidos en el programa periódico y sus frecuencias de difusión (y en consecuencia la cantidad de ancho de banda ocupada por el programa periódico) dependen tanto de la distribución de acceso observada por el servidor en el pasado como del ancho de banda necesario para transmitir cada elemento, de manera que la cantidad de ancho de banda asignada a cada modo va variando para adaptarse al patrón de acceso actual, siempre con el objetivo de minimizar el número total de plazos no cumplidos.

La frecuencia de transmisión de un elemento del programa periódico debe ser la mínima necesaria para satisfacer las restricciones temporales asociadas a dicho elemento. La longitud del ciclo de difusión y el ancho de banda asignado a cada modo, que llamaremos respectivamente *Periodic_Broadcast_Length* y *On_Demand_Length*, variarán de ciclo a ciclo en respuesta a cambios en las necesidades de los usuarios.

Debemos resaltar que el programa periódico se calcula al final de cada ciclo, mientras que la difusión bajo demanda se planifica conforme se reciben las peticiones de los usuarios. Es necesario asegurar que siempre existe al menos una pequeña fracción de ancho de banda para la difusión bajo demanda, para evitar que los elementos no incluidos en el programa periódico dejen de ser transmitidos totalmente.

El índice del programa periódico, que evita que la mayor parte de peticiones lleguen al servidor y en consecuencia la congestión del canal de subida, está intercalado con la información y se retransmite varias veces por ciclo, para reducir el tiempo necesario para que un usuario lo reciba.

Es importante observar que, debido a que los usuarios satisfechos con el programa periódico no envían peticiones al servidor, es imposible para el servidor conocer la distribución de acceso real exacta. De hecho, cuanto más ajustado sea el programa periódico a la distribución real, menos información tendrá el servidor sobre la misma.

Como consecuencia, el servidor no dispone de información suficiente para decidir si algún elemento debe ser retirado del programa periódico debido a que su frecuencia de acceso ha ido disminuyendo con el tiempo.

Este problema ha sido resuelto en el servidor AHB utilizando una técnica de muestreo, la cual básicamente consiste en dejar de transmitir un elemento del programa periódico durante una pequeña parte del ciclo, eliminando el elemento del índice del programa periódico consecuentemente. Esto resulta en la recepción de unas cuantas peticiones del elemento, mediante las cuales el servidor puede estimar la frecuencia de acceso real para el elemento en cuestión y decidir si el elemento debe ser retirado del programa periódico o permanecer en el mismo.

5.2 Clasificación de la información

En un instante de tiempo concreto, el servidor considera que un determinado elemento de información se encuentra en uno de los dos siguientes estados:

- **Periódico**: incluido en el programa periódico
- **Bajo Demanda**: no incluido en el programa periódico

Al final de cada ciclo de difusión, el servidor actualiza el estado de cada elemento, dependiendo de la estimación de la distribución de las frecuencias de acceso, y del ancho de banda requerido por cada elemento para ser incluido en el programa periódico.

Es lógico pensar que los elementos con alta frecuencia de acceso y bajo consumo de ancho de banda deberían ser difundidos periódicamente. Sin embargo, como el ancho de banda disponible es limitado, el objetivo del servidor es establecer el ancho de banda óptimo relativo destinado al programa periódico.

Intuitivamente, si deseamos maximizar la utilización del ancho de banda disponible, en situaciones de baja carga la mayoría de los elementos se encontrarán en el estado Bajo Demanda, mientras que en condiciones de sobrecarga un gran número de elementos estará en el estado Periódico, con el objetivo de satisfacer a una gran cantidad de usuarios mediante una sola transmisión.

Como ya hemos visto en el capítulo anterior, este problema tiene mucha semejanza con el conocido en la literatura algorítmica como el problema "fractional knapsack'', el cual es resuelto mediante una estrategia avara, seleccionando primero el elemento con el mayor "valor por kilo" (en nuestro caso mayor número de peticiones satisfechas potencialmente por unidad de ancho de banda), después el elemento con el segundo mayor "valor por kilo" y así sucesivamente hasta completar la capacidad del canal de bajada. Como ya demostramos con anterioridad, la solución avara es óptima en las condiciones concretas de nuestro problema.

Para conocer si existe un ahorro real de ancho de banda cuando un elemento es clasificado como Periódico, necesitamos saber el consumo de ancho de banda de ese elemento, el cual depende de la frecuencia de difusión. La frecuencia de difusión debe ser la mínima para asegurar que los plazos temporales asociados a las potenciales peticiones van a ser cumplidos.

Asumiremos que el servidor conoce a priori los valores de los posibles plazos permitidos para efectuar las solicitudes de cada elemento, lo cual es posible si a los usuarios sólo se les permite solicitar la información usando un conjunto de valores predeterminado. En cualquier caso, asumiremos que siempre conocemos el plazo mínimo con el que cada elemento es susceptible de ser solicitado.

Para cada elemento de información definimos su *prioridad* como el número de peticiones recibidas en el ciclo anterior para el elemento dividido por el ancho de banda requerido para difundir el elemento en el programa periódico. Con esto damos mayor prioridad a los elementos con mayor probabilidad de satisfacer a más usuarios **por unidad de ancho de banda**.

Para evitar la ausencia de transmisión de los elementos clasificados como Bajo Demanda en condiciones de sobrecarga, y para reservar el ancho de banda necesario para la transmisión del índice del programa periódico, definimos el parámetro *BW_Limit*, la máxima fracción de ancho de banda disponible para el programa periódico.

Por ejemplo, *BW_Limit* sería igual a 0.9 si queremos reservar al menos el 10% del ancho de banda para la difusión bajo demanda

y el índice en el peor caso. Este límite sólo se alcanzaría en fuertes condiciones de sobrecarga, ya que el número de elementos periódicos aumentaría considerablemente hasta ocupar todo el ancho de banda asignado al programa periódico.

A continuación pasamos a especificar el algoritmo adaptativo que es ejecutado por el servidor AHB al final de cada ciclo:

1. Calcular el consumo de ancho de banda de cada elemento. Supongamos que cada elemento tiene un cierto tamaño en páginas, o unidades de información a ser difundida. Si tenemos n elementos con tamaños (s_1, s_2, … , s_n) y los plazos temporales *mínimos* relativos asociados son (d_1, d_2, … , d_n), el ancho de banda requerido por el elemento i será[1]:

$$BWrequired_i = \frac{s_i + 1}{d_i + 1}$$

Este sería el ancho de banda necesario para transmitir el elemento i con la mínima frecuencia para satisfacer a todos los potenciales usuarios, según explicaremos con detalle en la sección siguiente.

2. Para cada elemento, calcular su prioridad, es decir, el número de usuarios que esperamos satisfacer por unidad de

[1] En principio podríamos pensar que:

$$BWrequired_i = \frac{s_i}{d_i}$$

Sin embargo, esto no garantiza que todos los plazos se cumplan para el elemento i, ya que la diferencia entre dos instancias consecutivas del elemento en el peor caso sería dos veces el plazo menos el tamaño del elemento. Esto ocurriría si en un ciclo del programa periódico el elemento fuera difundido al principio del programa y en el siguiente ciclo al final del mismo.

ancho de banda si el elemento fuera incluido en el programa periódico, considerando las peticiones recibidas en el ciclo anterior.

$$Priority_i = \frac{RequestsReceived_i}{BWrequired_i}$$

3. Considerar los elementos en orden de prioridad, empezando por el de máxima prioridad, para su posible inclusión en el programa periódico. Para cada elemento, comprobar si vale la pena incluirlo en el programa periódico, es decir, si el número de peticiones recibidas durante el último ciclo es superior al parámetro *CutoffThreshold*(i).

Este parámetro indica el número de veces por ciclo que el elemento i sería transmitido si fuera incluido en el programa periódico, y es igual a la longitud del ciclo de difusión dividida por el plazo asociado al elemento:

$$CutoffThreshold_i = \frac{BroadcastLength_i}{d_i}$$

La razón para la existencia de este valor es que si esperamos recibir menos peticiones que el número de copias del elemento que vamos a transmitir, malgastamos ancho de banda si lo incluimos en el conjunto Periódico.

4. Si la comprobación anterior es positiva, comprobar si el ancho de banda total requerido está por debajo del límite *BW_Limit* si el elemento fuera añadido al programa periódico. Si es así, el estado del elemento pasa a ser Periódico para el

siguiente ciclo, y volvemos al paso 3 para comprobar el siguiente elemento. Si no es así, continuamos al paso 5.

5. Hemos usado una cierta cantidad de ancho de banda para el programa periódico, nunca excediendo el límite *BW_Limit*. El resto es utilizado para satisfacer las peticiones recibidas por el canal de subida para elementos no incluidos en el programa periódico, es decir, servidos en modo bajo demanda.

A modo de resumen, ¿qué elementos son considerados parte del programa periódico en el siguiente ciclo de difusión? Los *m* elementos de mayor prioridad, que cumplan las siguientes dos condiciones:

$$\sum_{i=1}^{m} BWrequired_i \leq BWLimit$$

$$RequestsReceived_i > CutoffThreshold_i \qquad 1 \leq i \leq m$$

En la figura 5.1 mostramos el diagrama de flujo correspondiente al algoritmo adaptativo AHB, que será ejecutado por el servidor al final de cada ciclo de difusión.

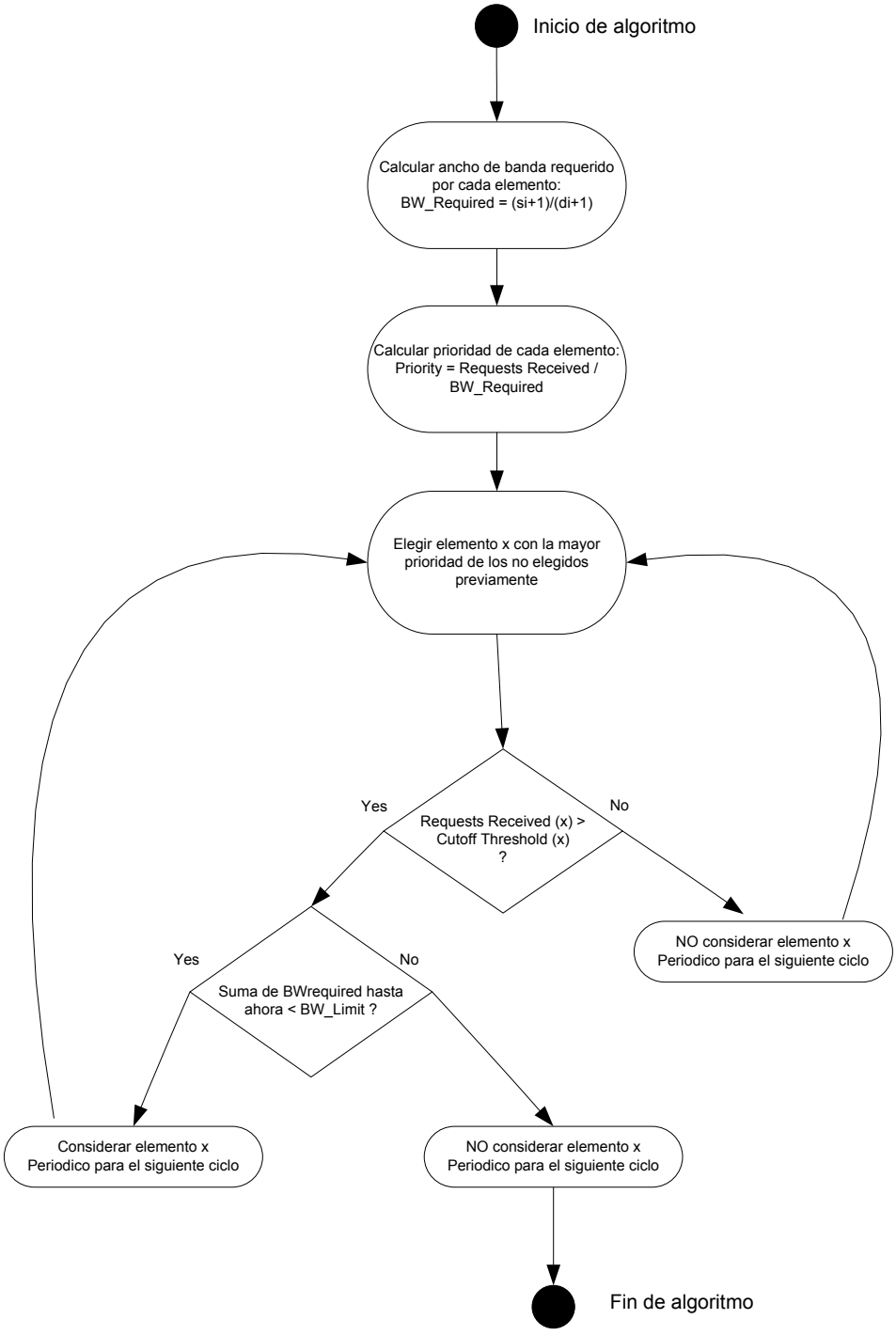

Figura 5.1. Diagrama de flujo del algoritmo AHB

5.3 Elaboración del programa periódico

Una vez que hemos seleccionado los elementos que van a formar parte del programa periódico en el siguiente ciclo, necesitamos un algoritmo que nos dé como resultado qué elemento concreto es transmitido en cada slot temporal.

En principio podríamos pensar que debe existir un algoritmo relativamente simple que garantice que se cumplen todos los plazos para los elementos elegidos para formar parte del programa periódico. En la práctica, debido a los posibles solapamientos, no es una tarea sencilla especificar tal algoritmo.

En el modelo de servidor híbrido BoD, se utiliza un algoritmo que, empleando el ancho de banda resultante de sumar los anchos de banda ocupados por los elementos integrantes del programa periódico cuando son transmitidos con un periodo igual al plazo asociado, trata de minimizar el total de plazos perdidos.

Sin embargo, como hemos citado anteriormente, en AHB utilizamos un algoritmo capaz de elaborar un programa periódico que garantiza que todos los plazos se cumplen para cada elemento del programa, con el coste de utilizar un ancho de banda total algo mayor que el del servidor BoD. Esto es posible si utilizamos un algoritmo de planificación basado en *pfair* [Baru94, Baru95].

A efectos ilustrativos, expondremos el algoritmo utilizado en BoD a continuación, para detallar el algoritmo correspondiente a AHB en el apartado 5.3.2.

5.3.1 Algoritmo para minimización de plazos perdidos

Este caso fue estudiado por el autor de esta tesis en [Xuan07], durante el desarrollo del servidor híbrido BoD. Básicamente el problema consiste en hallar la forma de distribuir un determinado número de copias de cada elemento de la forma más uniforme posible, es decir, tratando de minimizar los casos en los que la separación entre dos instancias consecutivas de un mismo elemento sea mayor que el valor de su plazo asociado.

Si tenemos un total de m elementos a integrar en el programa periódico, con tamaños en número de páginas (s_1, s_2, … , s_m) y plazos asociados (d_1, d_2, … , d_m), para minimizar el número de plazos perdidos en cada período del programa los elementos serán transmitidos un número de veces (n_1, n_2, … , n_m) respectivamente, siendo n_i = Mínimo_Común_Múltiplo{ d_1 , d_2 , … , d_m } /d_i .

Podemos afirmar que la longitud total de cada ciclo del programa periódico será:

$$L = \sum_{i=1}^{m} n_i s_i$$

Dado que las peticiones pueden llegar en cualquier instante temporal, el objetivo es colocar las n_i copias del elemento i lo más equidistantes posible dentro de la longitud total L del programa.

Esto significa que debemos minimizar la varianza del interespaciado. Si las n_i copias del elemento i son difundidas en los instantes a_1, a_2, … , a_{ni}, teniendo en cuenta que L/n_i es la

separación ideal para 2 transmisiones consecutivas del elemento i, debemos minimizar la siguiente fórmula:

$$\Delta_i = \sum_{j=2}^{n_i} \left(a_j - a_{j-1} - \frac{L}{n_i} \right)^2$$

Este problema es computacionalmente intensivo, por lo que podemos proponer una aproximación a la solución con mayor simplicidad de cálculo, como exponemos a continuación.

Elegimos un número B tal que B>=L, preferiblemente un múltiplo común de n_1, n_2, … , n_m, para que B/n_i sea un número entero y el error causado por redondeo pueda minimizarse. La idea es tratar de distribuir las ni copias de todos los elementos de una manera uniforme en B. Como B>=L, el programa contendrá slots vacíos, así que compactamos el mismo para hacerlo de longitud L. Los pasos para crear el programa son los siguientes:

1. Empezamos con una lista vacía de tamaño B, la plantilla para nuestro programa periódico. Llamamos c a la posición actual del slot a asignar en dicha lista.

2. Seleccionamos el elemento i, el siguiente a ser colocado en la lista (empezaremos por el primer elemento). Inicializamos c al primer slot de la lista.

3. Vamos colocando las ni copias del elemento i, de acuerdo con lo siguiente: si la posición actual c está ocupada, asignamos c al siguiente slot libre. Si se alcanza el final de la lista, comenzamos de nuevo por el principio y asignamos el primer slot libre a c. Seguidamente colocamos una copia de i

en el slot c y sucesivos libres, dependiendo del tamaño del elemento s_i. Finalmente incrementamos c en B/n_i.

4. Repetimos los pasos 2 y 3 hasta haber colocado todas las copias de los m elementos.

5. Compactamos la lista eliminando los slots vacíos.

La razón de ser del paso 3 es que, idealmente, para el elemento i, dos copias adyacentes del mismo deberían estar separadas por B/n_i. En el caso de que el slot ideal esté ocupado por algún elemento que haya sido colocado previamente, intentamos buscar el lugar libre más cercano. Como B es un número grande, estadísticamente encontraremos el siguiente sitio libre con mayor rapidez.

Como ejemplo de un programa calculado con el algoritmo, consideremos los elementos i_1, i_2 e i_3. Cada elemento viene caracterizado por una dupla (tamaño, plazo relativo). En nuestro caso utilizaremos las duplas i_1=(1, 2), i_2=(2, 7), i_3=(1, 14), como entrada al algoritmo.

Siendo el mínimo común múltiplo de los plazos MCM = 14, el número de veces que cada elemento será transmitido en cada período del programa es n_1= 14/d_1= 7, n_2 = 14/ d_2= 2 y n_3 = 14/ d_3 = 1. En consecuencia, L = 7x1 + 2x2 + 1x1 = 12.

El programa periódico se crea como sigue: elegimos B = 28 (múltiplo de n_i), tomamos el primer elemento i_1, inicializamos c al primer slot de la lista, colocamos una copia de i_1, incrementamos c

en $B/n_1 = 28/7 = 4$, colocamos la siguiente copia de i_1, y así sucesivamente hasta colocar las 7 copias de i_1.

A continuación tomamos i_2, y como el primer slot de la lista está ya asignado, colocamos la primera copia en los slots 1 y 2. Incrementamos c en $B/n_2 = 28/2 = 14$, e intentamos colocar la segunda copia en los slots 15 y 16. Como el slot 16 de la lista está ya asignado, colocamos la primera página en el slot 15 y la segunda página en el 17. Ya hemos colocado las 2 copias de i_2.

Por último tomamos i_3, y como el primer, segundo y tercer slot de la lista están ya asignados, colocamos la primera y única copia en el siguiente slot.

El resultado final del algoritmo es el siguiente:

Elemento	Slots asignados
i_1	0,4,8,12,16,20,24
i_2	1,2,15,17
i_3	3

Tras compactar la lista eliminando los slots vacíos, el programa periódico generado es el siguiente:

Programa periódico											
1	2	2	3	1	1	1	2	1	2	1	1

Podemos observar que todas las peticiones que lleguen para el elemento i1 durante el segundo y tercer slots perderán sus

respectivos plazos. Es decir, el programa periódico resultante NO garantiza la ausencia total de plazos perdidos.

5.3.2 Algoritmo con garantía de ausencia de plazos perdidos

Este es el algoritmo escogido para la elaboración del programa periódico en nuestro servidor híbrido AHB. La familia de algoritmos basados en *pfair* [Baru94, Baru95] calcula eficientemente un programa periódico en el que se garantiza que cualquier subsecuencia de d_i páginas contiene al menos s_i páginas correspondientes al elemento i. El programa periódico resultante proporciona una progresión justa en la asignación del recurso (en nuestro caso el canal de bajada) al conjunto de elementos en cuestión.

Utilizando un algoritmo de este tipo, todos los plazos para los elementos integrantes del programa periódico se cumplirán, ya que para el elemento i se transmitirán s_i páginas de cada d_i. Asumimos que los usuarios pueden reordenar las páginas recibidas localmente.

El problema de planificación al que nos enfrentamos puede ser formalmente definido como sigue:

Dado un conjunto {(s_1, d_1), (s_2, d_2), …, (s_m, d_m)} de pares ordenados de enteros positivos, determinar una secuencia sobre los símbolos {1,2,3, …, m} tal que, repitiendo la secuencia periódicamente, para cada i tal que 1<=i<=m, cualquier subsecuencia de d_i símbolos consecutivos contenga al menos s_i i's.

A continuación describimos el algoritmo utilizado para generar la secuencia deseada. Está basado en el algoritmo *PinOpt* [Baru97c].

Previamente establecemos las siguientes definiciones:

- *allocated(i,t)* = número de slots (páginas o unidades de transmisión) asignados al elemento i en el intervalo [0,t)

- El elemento i estará *compitiendo* en el slot t si puede recibir el recurso sin haber recibido el recurso demasiadas veces, es decir, si la siguiente condición es verdadera:

$$\left\lceil \frac{allocated(i,t)}{BWrequired_i} \right\rceil \leq t$$

- El pseudoplazo (*pseudodeadline*) del elemento i en el slot t se define como:

$$pseudodeadline(i,t) = \left\lceil \frac{allocated(i,t)+1}{BWrequired_i} \right\rceil$$

Los dos pasos que conforman el algoritmo son:

Paso 1: Añadir un elemento ficticio (s_{m+1}, d_{m+1}) tal que:

$$\sum_{i=1}^{m+1} BWrequired_i = 1$$

Es importante recordar que:

$$\sum_{i=1}^{m} BWrequired_i \leq BWLimit$$

Siendo:

$$BWrequired_i = \frac{s_i + 1}{d_i + 1}$$

Los slots correspondientes al elemento ficticio m+1 se usarán para transmitir los elementos no incluidos en el programa periódico (es decir, en el estado Bajo Demanda) y las copias del índice.

Paso 2: Planificar el nuevo conjunto de pares asignando cada slot al elemento que posea el menor pseudo-plazo dentro de los elementos que estén compitiendo por el slot. En caso de igualdad entre varios elementos puede elegirse cualquiera de ellos.

El programa resultante es justo en el sentido de que garantiza que a cada elemento i (para i entre 1 y m+1) se le asigna el recurso exactamente una vez durante el intervalo:

$$\left[\left\lceil \frac{j}{BWrequired_i} \right\rceil, \left\lceil \frac{j+1}{BWrequired_i} \right\rceil\right)$$

Esta asignación del recurso se garantiza para cada entero j >= 0, siempre asumiendo que se cumple la igualdad establecida en el paso 1.

Este algoritmo puede ser implementado de forma eficiente utilizando una estructura de datos llamada "*heap-of-heaps*", con una complejidad O(log(m)) por slot [Mok88]. El período del programa generado es el mínimo común múltiplo de los plazos, y ésta será la longitud del ciclo.

Como ejemplo de un programa calculado con el algoritmo, consideremos los elementos i_1, i_2 e i_3, caracterizados por {(1, 4), (2, 7), (1, 14)}, como entrada al algoritmo.

Para aplicar nuestro algoritmo a los elementos anteriores, calculamos primero los anchos de banda requeridos por cada elemento:

Elemento	BWrequired
i_1	2/5 = 0.4
i_2	3/8 = 0.375
i_3	2/15 = 0.133
TOTAL	0.908

Paso 1: Añadimos un elemento ficticio i_4 tal que la suma de anchos de banda requeridos sea la unidad. Esto significa que asignamos un ancho de banda de 0.092 al elemento ficticio i_4.

Paso 2: Vamos asignando cada slot al elemento con menor pseudoplazo de los que estén compitiendo por el slot. En caso de igualdad elegimos el elemento con el menor índice.

Podemos ver el programa periódico generado por la ejecución del algoritmo en la siguiente tabla:

Tabla 5.1. Programa periódico generado por el algoritmo con garantía de ausencia de plazos perdidos

Slot	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**
$allocated(1,t)$	0	1	1	1	2	2	3	3	3	4	4	5	5	5
$allocated(2,t)$	0	0	1	1	1	2	2	3	3	3	4	4	4	5
$allocated(3,t)$	0	0	0	1	1	1	1	1	1	1	1	1	2	2
$allocated(4,t)$	0	0	0	0	0	0	0	0	1	1	1	1	1	1
$\left\lceil \frac{allocated(1,t)}{BWrequired_1} \right\rceil$	0	3	3	3	5	5	8	8	8	10	10	13	13	13
$\left\lceil \frac{allocated(2,t)}{BWrequired_2} \right\rceil$	0	0	3	3	3	6	6	8	8	8	11	11	11	14
$\left\lceil \frac{allocated(3,t)}{BWrequired_3} \right\rceil$	0	0	0	8	8	8	8	8	8	8	8	8	15	15
$\left\lceil \frac{allocated(4,t)}{BWrequired_4} \right\rceil$	0	0	0	0	0	0	0	0	11	11	11	11	11	11
i_1 *compitiendo*	Sí	No	No	Sí	No	Sí	No	No	Sí	No	Sí	No	No	Sí
i_2 *compitiendo*	Sí	Sí	No	Sí	Sí	No	Sí	No	Sí	Sí	No	Sí	Sí	No
i_3 *compitiendo*	Sí	Sí	Sí	No	No	No	No	No	Sí	Sí	Sí	Sí	No	No
i_4 *compitiendo*	Sí	Sí	Sí	Sí	Sí	Sí	Sí	Sí	No	No	No	Sí	Sí	Sí
$pseudodeadline(1,t)$	3	5	5	5	8	8	10	10	10	13	13	15	15	15
$pseudodeadline(2,t)$	3	3	6	6	6	8	8	11	11	11	14	14	14	16
$pseudodeadline(3,t)$	8	8	8	15	15	15	15	15	15	15	15	15	23	23
$pseudodeadline(4,t)$	11	11	11	11	11	11	11	11	22	22	22	22	22	22
Programa	$\mathbf{i_1}$	$\mathbf{i_2}$	$\mathbf{i_3}$	$\mathbf{i_1}$	$\mathbf{i_2}$	$\mathbf{i_1}$	$\mathbf{i_2}$	$\mathbf{i_4}$	$\mathbf{i_1}$	$\mathbf{i_2}$	$\mathbf{i_1}$	$\mathbf{i_3}$	$\mathbf{i_2}$	$\mathbf{i_1}$

Slot	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**
$allocated(1,t)$	6	6	7	7	7	8	8	9	9	9	10	10	11	11
$allocated(2,t)$	5	6	6	7	7	7	8	8	8	9	9	10	10	10
$allocated(3,t)$	2	2	2	2	2	2	2	2	3	3	3	3	3	4
$allocated(4,t)$	1	1	1	1	2	2	2	2	2	2	2	2	2	2
$\left\lceil \frac{allocated(1,t)}{BWrequired_1} \right\rceil$	15	15	18	18	18	20	20	23	23	23	25	25	28	28
$\left\lceil \frac{allocated(2,t)}{BWrequired_2} \right\rceil$	14	16	16	19	19	19	22	22	22	24	24	27	27	27
$\left\lceil \frac{allocated(3,t)}{BWrequired_3} \right\rceil$	15	15	15	15	15	15	15	15	23	23	23	23	23	30
$\left\lceil \frac{allocated(4,t)}{BWrequired_4} \right\rceil$	11	11	11	11	22	22	22	22	22	22	22	22	22	22
i_1 *compitiendo*	No	Sí	No	No	Sí	No	Sí	No	No	Sí	No	Sí	No	No
i_2 *compitiendo*	Sí	No	Sí	No	No	Sí	No	No	Sí	No	Sí	No	No	Sí
i_3 *compitiendo*	No	Sí	Sí	Sí	Sí	Sí	Sí	Sí	No	Sí	Sí	Sí	Sí	No
i_4 *compitiendo*	Sí	Sí	Sí	Sí	No	No	No	No	Sí	Sí	Sí	Sí	Sí	Sí
$pseudodeadline(1,t)$	18	18	20	20	20	23	23	25	25	25	28	28	30	30
$pseudodeadline(2,t)$	16	19	19	22	22	22	24	24	24	27	27	30	30	30
$pseudodeadline(3,t)$	23	23	23	23	23	23	23	23	30	30	30	30	30	38
$pseudodeadline(4,t)$	22	22	22	22	33	33	33	33	33	33	33	33	33	33
Programa	$\mathbf{i_2}$	$\mathbf{i_1}$	$\mathbf{i_2}$	$\mathbf{i_4}$	$\mathbf{i_1}$	$\mathbf{i_2}$	$\mathbf{i_1}$	$\mathbf{i_3}$	$\mathbf{i_2}$	$\mathbf{i_1}$	$\mathbf{i_2}$	$\mathbf{i_1}$	$\mathbf{i_3}$	$\mathbf{i_2}$

Podemos comprobar como en cada 4 slots consecutivos el elemento 1 se transmite como mínimo 1 vez, en cada 7 slots el elemento 2 se transmite como mínimo 2 veces, y cada 14 slots el elemento 3 se transmite como mínimo 1 vez.

Los slots asignados al elemento 4 se emplearían en la transmisión del índice del programa periódico y de los elementos Bajo Demanda.

5.4 Planificación de la difusión bajo demanda

Las peticiones que llegan al servidor pertenecen a usuarios no satisfechos con el programa periódico. En teoría deben corresponder con información demandada poco frecuentemente o con alto consumo de ancho de banda. Las peticiones son insertadas en una cola de prioridades, de la cual el servidor extraerá una determinada petición para ser servida cada vez que disponga de un slot libre asignado a difusión bajo demanda.

Las peticiones serán satisfechas por el servidor de acuerdo con sus prioridades, utilizando el ancho de banda sobrante del programa periódico, es decir, los slots reservados al elemento ficticio m+1 en el algoritmo de la sección anterior, teniendo en cuenta también el consumo de ancho de banda del índice del programa periódico.

El ancho de banda disponible para este modo depende de la longitud del programa periódico, es decir, de la carga ofrecida por los usuarios en cada momento. Debemos recordar que siempre existe al menos una fracción de ancho de banda reservada para este modo (y para el índice), dada por (1-*BW_Limit*).

El servidor necesita una política de planificación para establecer un orden en el servicio de las peticiones. La política non-preemptive Earliest Deadline First (EDF, plazo más cercano primero sin desalojo) [Liu73] ha sido extensivamente empleada en sistemas de tiempo real para el cumplimiento de plazos temporales, ya que está demostrado que es una solución óptima para el problema de

minimización de plazos perdidos en planificación de tareas para un servidor sin desalojo [Liu73].

En EDF la prioridad es inversamente proporcional al plazo temporal relativo de la petición, es decir, se sirve primero la de menor plazo temporal relativo. Las peticiones cuyo plazo no puede ser satisfecho simplemente se desestiman. Podemos describir el funcionamiento de EDF en forma de pseudocódigo como sigue:

```
EDF(Target, t)
{
      // Target = peticiones a planificar
      // t = slot más cercano en el que se puede planificar
      // Scheduled = peticiones en Target que han sido planificadas
      // Di = información demandada por la petición i
      // Ti = plazo asociado a la petición i

      // inicializamos
      Scheduled = 0;

      // ordenamos Target en orden de plazos crecientes
      Sort(Target);

      while ( Target != Empty ) {
            // Elegimos petición con plazo más cercano
            i = GetFirstRequest(Target);

            // Comprobamos si i es planificable
            if ( t + Tamaño(Di) <= Ti) {
                  // Reservamos Slots para transmitir i
                  ReserveSlotInterval(t,t + Tamaño(Di));
                  t = t + Tamaño(Di);

                  // Añadimos i a Scheduled
                  AddToScheduled(i);
            }
            else {
                  // Rechazamos la petición i
```

```
                Reject(i);
            }
        }
}
```

Utilizando esta política no tenemos en cuenta ni la frecuencia de acceso a la información ni el ancho de banda requerido por cada elemento, solamente los plazos asociados.

EDF ha demostrado ser una política de planificación óptima [Liu73] para sistemas que no utilizan difusión. Sin embargo, es fácil ver que si planificamos una transmisión por petición recibida, EDF obtendrá unas prestaciones muy bajas en el caso de múltiples peticiones para el mismo elemento. EDF no tiene en cuenta que no es necesario realizar múltiples transmisiones de la misma información si con la primera transmisión se satisfacen los plazos de las siguientes peticiones.

La eficiencia de la política de planificación EDF puede ser aumentada si agrupamos varias peticiones para un mismo elemento (EDF_BATCHING), ya que podemos satisfacerlas con una única transmisión, siempre que cumplamos los plazos asociados a las mismas.

Cuando el servidor recibe una petición, primero comprueba si en la cola de prioridades existe otra petición del mismo elemento y si el plazo de la misma no es superior al de la recién recibida, en cuyo caso no es necesario planificar la nueva. Esta técnica de agrupamiento produce un importante ahorro de ancho de banda en condiciones de sobrecarga [Xuan97].

Esta aproximación, aunque sirve un grupo de peticiones con una sola transmisión, no introduce retardos adicionales para ningún cliente, y la sobrecarga con respecto a EDF puro es despreciable.

Como alternativa a EDF_BATCHING, también podemos emplear el algoritmo SIN-α [Xu06] para el canal de difusión bajo demanda. Al contrario que EDF, este algoritmo tiene en cuenta tanto los plazos relativos como la popularidad de la información al seleccionar la petición a ser atendida.

5.5 Estimación de la distribución de frecuencias de acceso

La clave para obtener una alta eficacia de nuestro sistema híbrido AHB es la identificación de los elementos que consumen menor ancho de banda por petición satisfecha, de modo que sólo deban ser servidos bajo demanda los elementos poco demandados y/o que precisen una gran frecuencia relativa de difusión.

Recordemos que las necesidades reales de los usuarios no son caracterizables ni predecibles a priori, dada la naturaleza dinámica de los mismos. En general, en determinadas situaciones "pico" se producirán aumentos considerables de la demanda total.

Como ya hemos mencionado anteriormente, la transmisión del programa periódico combinado con el índice disminuye enormemente el número de peticiones que deben ser atendidas por el servidor, aumentando la escalabilidad del mismo. Desafortunadamente, esto provoca que el servidor no reciba información actualizada sobre el número de usuarios satisfechos por

el programa periódico, información necesaria para calcular la prioridad real de los elementos a incluir en el programa periódico al final de cada ciclo.

Imaginemos que el servidor ha recibido un gran número de peticiones para un elemento durante un ciclo y al final del ciclo decide incorporarlo al programa periódico. Dado que en el siguiente ciclo todos los plazos correspondientes a peticiones del elemento en cuestión serán satisfechos, al final del siguiente ciclo el servidor AHB no habrá recibido ninguna petición del mismo, con lo cual no puede conocer el número de usuarios satisfechos durante el ciclo.

De forma paradójica, cuanto mejor sea la adaptación del sistema a la población de usuarios, menor será el número de peticiones recibidas, y menor capacidad de adaptación tendrá el sistema frente a cambios en la demanda.

Las necesidades reales de los usuarios no pueden ser caracterizadas ni estimadas basándose en datos pasados debido a la naturaleza dinámica de los mismos (basta pensar en situaciones de emergencia que pueden causar cambios abruptos en la demanda de información). Además, los usuarios que reciben información de un canal de difusión son pasivos en el sentido de que no comunican al servidor de forma directa que han recibido la información correcta.

Para solucionar este problema hemos incorporado una técnica de muestreo inspirada en el método usado en [Stat97]. Este método define una temperatura para cada elemento de la base de datos, que se corresponde con su tasa de peticiones. De manera intuitiva, cada elemento puede encontrarse en un instante determinado en uno de los siguientes 3 estados:

- **Vapor**: Elementos fuertemente demandados que forman parte del programa periódico

- **Líquido**: Elementos moderadamente demandados, su tasa de demanda no justifica su inclusión en el programa periódico

- **Sólido** (hielo): Elementos que no han sido solicitados durante un tiempo considerable, es decir, su temperatura ha descendido prácticamente a cero

El servidor determina de forma dinámica el estado de cada elemento de la base de datos. La parte más complicada de este esquema es la distinción de forma certera entre los estados Vapor y Líquido.

Cuando un elemento es añadido al programa periódico, pasa al estado "Vapor", y el número de peticiones recibidas en el ciclo anterior es almacenado. Al final del siguiente ciclo, este número de peticiones es disminuido artificialmente multiplicándolo por un factor de enfriamiento *Cooling_Factor*, típicamente en el rango 0.5-0.9, es decir, el elemento se va enfriando con el paso de los ciclos. Este número de peticiones disminuido ficticiamente es usado para calcular la prioridad estimada del elemento.

Después de unos pocos ciclos, esta prioridad estimada va decreciendo hasta el punto de que el elemento no debe ser incluido en el programa periódico, pasando el elemento al estado "Líquido". En este punto, antes de sacar al elemento definitivamente del programa, la técnica de muestreo es usada para estimar la frecuencia de acceso real del elemento.

La técnica de muestreo consiste en que el servidor incluye el elemento en el programa periódico, pero no durante todo el ciclo. Durante un corto intervalo de tiempo al final del ciclo, llamado tiempo de muestreo (*Sampling_Time*), el servidor no transmite el elemento (el índice del programa periódico es actualizado para reflejar este cambio), recibiendo un número pequeño de peticiones y pudiendo así estimar la frecuencia de acceso real del mismo.

Llamaremos a este pequeño número de peticiones que esperamos recibir durante el muestreo *Expected_Sample_Size*. El tiempo de muestreo que debemos utilizar es fácil de calcular, conociendo el número de peticiones recibidas en el ciclo anterior a la inclusión del elemento en el programa periódico (ciclo j):

$$Sampling\_Time_i = \frac{Broadcast\_Cycle\_Length(j) \cdot Expected\_Sample\_Size}{Requests\_Received_i(j)}$$

El número estimado de peticiones en el ciclo de muestreo (ciclo k) se calcula utilizando el número real de peticiones recibidas en el muestreo:

$$Estimated\_Requests_i(k) = \frac{Broadcast\_Cycle\_Length(k) \cdot Actual\_Sample\_Size_i}{Sampling\_Time_i}$$

Esta estimación sirve para calcular la prioridad estimada del elemento en cuestión y tomar la decisión final sobre si debe permanecer como Periódico o debe pasar a ser Bajo Demanda, es decir, si debe ascender del estado "Líquido" a "Vapor" o por el contrario descender de "Líquido" a "Sólido".

El tamaño de la ventana de muestreo debe ser seleccionado cuidadosamente, ya que es un factor crítico para asegurar la eficacia del muestreo. Si pensamos en una ventana muy pequeña, la precisión será baja; por el contrario, un gran tamaño de ventana producirá una enorme sobrecarga en el sistema, ocasionando en la práctica el mismo problema que la técnica de muestreo trata de resolver.

Un valor bajo de *Cooling_Factor* nos asegura una adaptación rápida frente a cambios en la distribución de acceso, pero al mismo tiempo implica que los elementos del programa periódico van a ser muestreados con mayor frecuencia, incrementando el número de peticiones recibidas y en consecuencia la sobrecarga del sistema. Por el contrario, valores altos ocasionan menor necesidad de muestreo pero comprometen la adaptabilidad del sistema.

Existe un compromiso en los valores elegidos para *Cooling_ Factor* y *Expected_Sample_Size*. La sobrecarga producida por valores bajos de *Cooling_Factor* puede reducirse seleccionando un *Expected_Sample_Size* bajo, pero esto acarrea necesariamente una pérdida de precisión en la estimación de peticiones recibidas para todo el ciclo de difusión.

Capítulo 6:

Simulación del modelo AHB

Con el objetivo de evaluar cuantitativamente las prestaciones de nuestro modelo de servidor híbrido adaptativo, hemos llevado a cabo varios experimentos utilizando simulación.

Hemos simulado un entorno de comunicación asimétrico que consta de un único servidor y un número ilimitado de usuarios móviles que pueden encontrarse en la zona de influencia del servidor.

6.1 Descripción del simulador

El servidor posee una base de datos con n elementos, no necesariamente del mismo tamaño. Los usuarios han sido modelados por medio de un generador de carga parametrizable, que produce un flujo de demandas de información.

El tiempo entre demandas de información ha sido modelado mediante una distribución exponencial, y en consecuencia el número de demandas sigue una distribución de Poisson, es decir, el número de demandas que ocurren en un determinado intervalo temporal es independiente del que ocurre en cualquier otro intervalo (se dice que la distribución de Poisson no tiene memoria) y la tasa de demandas tiene un ritmo constante, es decir, la probabilidad de que se produzca un evento en un intervalo de tiempo muy corto es proporcional a la longitud de dicho intervalo.

Este generador de carga tiene una serie de parámetros de entrada tales como tiempo total de simulación, tiempo medio entre demandas, número de elementos, valores de plazos asociados a las demandas, tipo de distribución a utilizar, etc., en función de los cuales genera ficheros de carga que se utilizan como entrada de datos del simulador.

Cada fichero generado contiene una línea por demanda de información, la cual incluye el instante temporal en que se ha producido la demanda, el plazo temporal asociado a la misma, el tamaño en páginas y el identificador del elemento demandado. A continuación mostramos como ejemplo una porción de un fichero generado:

```
##   the data lines contains the following info:
##  1. time stamp (current clock, i.e., number of ticks)
##  2. data deadline ( 0 if no deadline given -- means best effort)
##  3. data size   (number of packets)
##  4. data index  (used to identify data)

      ## data:
             0      600      1      9
             2      302      1    887
             4      904      1     37
             5      905      1     46
             6      306      1    910
             6      906      1    437
             7      907      1      3
             7      907      1      3
            11      611      1   2601
            13      913      1     37
            14      614      1    780
            16      616      1     19
            16      316      1     13
            19      919      1   4490
            20      320      1     18
            21      921      1      0
            22      322      1    223
            22      322      1     59
            22      922      1     91
            24      324      1    150
            25      325      1    687
            31      331      1     14
            32      932      1      0
            35      335      1    191
            36      336      1     13
```

Nótese que hablamos de demandas de información y no de peticiones expresas de información, ya que en servidores híbridos la gran mayoría de los usuarios obtendrán la información demandada del programa periódico y no realizarán petición alguna. De hecho, el

objetivo de un servidor híbrido adaptativo debe ser ajustar el programa periódico a la distribución real, para minimizar el número de peticiones que han de ser atendidas.

El simulador comienza la ejecución en tiempo cero y va leyendo las demandas de información del fichero generado a medida que el tiempo de simulación alcanza los instantes temporales en que se han producido las demandas.

La granularidad o unidad de tiempo empleada en nuestro simulador es el tiempo necesario para la difusión de una unidad de información (página) en el canal de bajada. Asumimos que el elemento más pequeño de la base de datos posee un tamaño igual a una página, y por tanto sería transmitido en el canal de bajada en una unidad de tiempo.

De esta forma, no es necesario especificar la capacidad absoluta del canal de bajada (por ejemplo: 1 Mbps), ya que en un caso real esto afectaría al tiempo necesario para la difusión de una página, y estamos tomando esa cantidad como la unidad de tiempo básica en el simulador. Es decir, la capacidad absoluta del canal de bajada va implícita al expresar el tiempo entre demandas de información en unidades de número de páginas.

Cada demanda de información es procesada de acuerdo al índice del elemento solicitado. Si el índice pertenece al programa periódico, la demanda es ignorada, ya que realmente no se ha producido una petición expresa. Si el índice no pertenece al programa periódico, la demanda es en realidad una petición y el servidor la inserta en una cola de prioridad de acuerdo a la política de planificación empleada.

De igual forma, el simulador tiene en cuenta si se ha sobrepasado la capacidad máxima del canal de subida, en cuyo caso calcula si la petición debe ser ignorada a causa del efecto de las colisiones que se producen en el canal de subida.

Hemos utilizado dos métricas para evaluar las prestaciones de cada modelo de servidor:

1. El porcentaje total de plazos perdidos, es decir, el porcentaje de casos donde un usuario es incapaz de obtener la información demandada dentro del plazo asociado (ya sea del programa periódico o bajo demanda).

2. El número total de peticiones expresas que son recibidas por el servidor, correspondiendo a usuarios no satisfechos con el programa periódico. Con esto evaluaremos el efecto de saturación del canal de subida.

Hemos modelado dos distribuciones diferentes de frecuencias de acceso (ver figura 6.1):

- **Uniforme de dos niveles**: Combina dos frecuencias de acceso constantes en la misma distribución. Los primeros 20 elementos son accedidos con un 90% de probabilidad (cada elemento de estos 20 es accedido con un 4,5% de probabilidad), y el resto de elementos es accedido en su conjunto con un 10% de probabilidad (para 1000 elementos, cada elemento tendría un 0,01% de probabilidad de acceso).

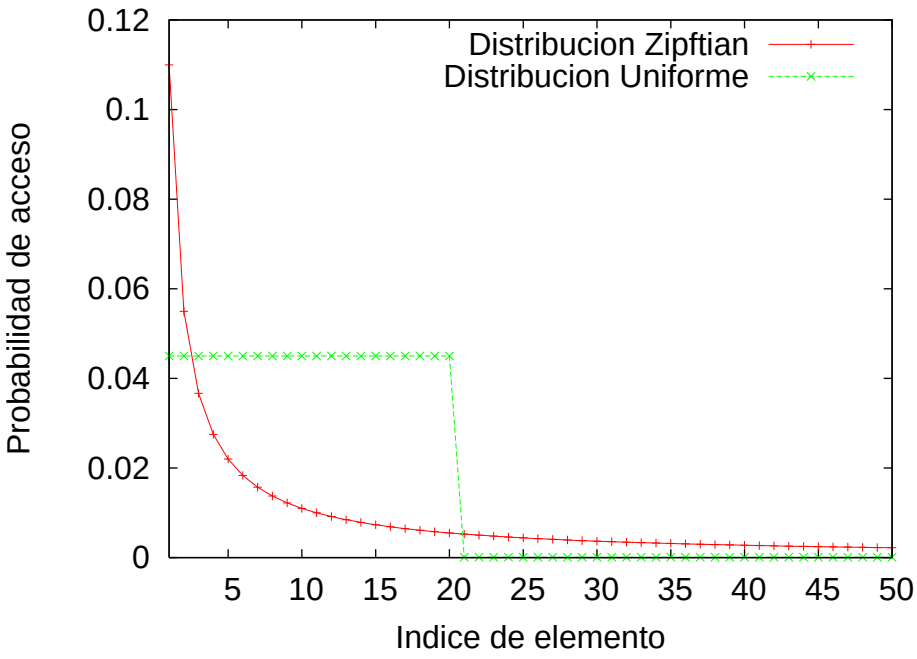

Figura 6.1. Distribuciones de frecuencias de acceso a la información

- **Zipftian** [Zip49], donde la probabilidad de acceder al elemento i es proporcional a (1/i). Como se aprecia en la figura 6.1, en esta distribución, existe mucha mayor probabilidad de acceso a los primeros elementos que a los últimos.

La primera distribución la hemos elegido con el objetivo de tener una distinción abrupta entre elementos muy demandados y poco demandados. La segunda distribución es un caso mucho más próximo a la realidad, de hecho varias investigaciones (por ejemplo [Bres99]) concluyen que las peticiones de páginas web en Internet siguen generalmente una distribución Zipftian.

Los experimentos han sido llevados a cabo utilizando versiones estáticas y dinámicas de las distribuciones. En las versiones dinámicas se han cambiado los elementos más fuertemente demandados varias veces durante la simulación.

Debemos resaltar que el rango de variación de la carga ofrecida ha sido diseñado para que el servidor trabaje en condiciones de sobrecarga durante todas las simulaciones.

La distribución de los plazos es uniforme, permitiendo un pequeño conjunto de valores. Para un elemento dado, uno de estos valores está permanentemente asociado con él, y representa el mínimo plazo con el que puede ser solicitado.

El índice está intercalado con los datos y su consumo de ancho de banda representa aproximadamente un 1% del total. El índice es transmitido cada 10 unidades de tiempo y consume una décima parte de una unidad de tiempo. El tiempo de espera para recibir el índice ha sido tenido en cuenta en los experimentos.

En cuanto a la técnica de muestreo utilizada, como ya apuntamos en el capítulo anterior, existe un compromiso en los valores elegidos para los parámetros *Cooling_Factor* y *Expected_Sample_Size*. Cuanto menor sea el valor del primero, la adaptación será más rápida, pero los elementos periódicos van a ser muestreados con mayor frecuencia y esto incrementa el número de peticiones recibidas. Podemos pensar en tener un *Expected_Sample_Size* lo más bajo posible, pero esto resulta en una pérdida de precisión en la estimación de las peticiones recibidas en el ciclo completo.

La máxima capacidad del canal de subida sólo puede alcanzarse cuando la carga ofrecida es exactamente la máxima capacidad. Si se supera esa carga, la capacidad efectiva disminuye, debido al aumento de las colisiones, ya que el protocolo de acceso al canal es basado en contención [Tann96]. En nuestro simulador, hemos modelado el efecto de la contención asumiendo que tras alcanzar la máxima capacidad del canal, la capacidad real va disminuyendo linealmente al aumentar la carga.

6.2 Comparación con modelos híbridos

A continuación vamos a comparar nuestro modelo AHB con el modelo *BoD* (Híbrido No Adaptativo) [Xuan97]. Recordamos que éste es el único modelo híbrido que considera peticiones con plazos asociados.

En el modelo BoD una fracción fija del ancho de banda se reserva para el programa periódico. Los elementos que serán transmitidos periódicamente se deciden a priori. En este caso hemos elegido los primeros i elementos (1, 2, 3, etc.) que caben en una fracción fija del ancho de banda para formar parte del programa periódico. El programa periódico es fijo y es calculado usando el algoritmo para minimizar el número de plazos perdidos detallado en el capítulo anterior.

En este caso hemos reservado en BoD un porcentaje de ancho de banda para el programa periódico de manera tal que los 20 primeros elementos formen parte del mismo. De esta forma, el programa periódico de BoD encaja totalmente con la carga ofrecida en distribución uniforme de dos niveles.

En esta comparación hemos utilizado en nuestro servidor AHB el mismo algoritmo para la elaboración del programa periódico que el utilizado por BoD, es decir, el algoritmo para minimización de plazos perdidos detallado en el capítulo anterior. De la misma forma, hemos utilizado EDF con agrupamiento para el modo de difusión bajo demanda en AHB, ya que es la misma política utilizada por BoD.

La siguiente tabla muestra los parámetros más importantes de la simulación y los valores elegidos para ellos. Estos valores representan un entorno de comunicación asimétrico de manera realista y son similares a los elegidos en [Stat97, Xuan97].

Tabla 6.1. Parámetros de simulación, modelos híbridos

Parámetro	**Valor**
Número de elementos	1000
Tamaño de cada elemento	10 páginas (=10 unidades de tiempo)
Plazos relativos	100, 200 o 500 unidades de tiempo
Tiempo de simulación	100000 unidades de tiempo
BW_Limit	0.9
Cooling_Factor	0.9
Expected_Sample_Size	5 peticiones
Capacidad del canal de subida	1 petición por unidad de tiempo
Rango de carga soportado	[0.1-4] elementos por unidad de tiempo

6.2.1 Distribución estática de entrada

Nuestro primer experimento utiliza una distribución estática uniforme de dos niveles. Como ya hemos indicado, el programa periódico utilizado en BoD se ajusta perfectamente a los elementos más fuertemente demandados de la distribución.

Como podemos observar en la figura 6.2, el algoritmo utilizado en AHB se comporta de manera ligeramente inferior a BoD, la diferencia es mayor cuando la carga ofrecida es baja, debido principalmente a la técnica de muestreo de AHB, ya que el número de peticiones de los elementos del programa periódico no es muy alto y el tiempo de muestreo será una fracción alta de la longitud del ciclo.

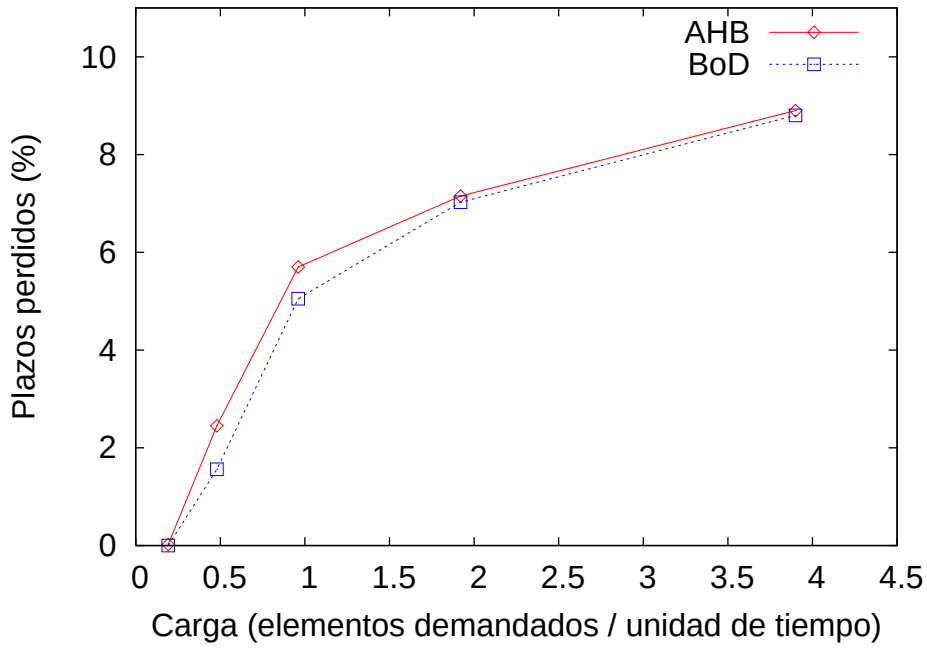

Figura 6.2. Distribución estática uniforme, plazos perdidos

La figura 6.3 muestra el número de peticiones que deben ser atendidas en el servidor. Las diferencias son mínimas, podemos ver la correlación que se produce entre el porcentaje de plazos perdidos y el número de peticiones que deben ser planificados por el servidor.

Como observación, apuntar que al encontrarnos ante modelos híbridos, tanto en AHB y BoD sólo deben ser planificadas aproximadamente el 10% de peticiones del total de la carga. El máximo flujo de peticiones es 4 elementos demandados por unidad de tiempo, lo cual sobrepasa en cuatro veces la capacidad máxima del canal (1 petición por unidad de tiempo).

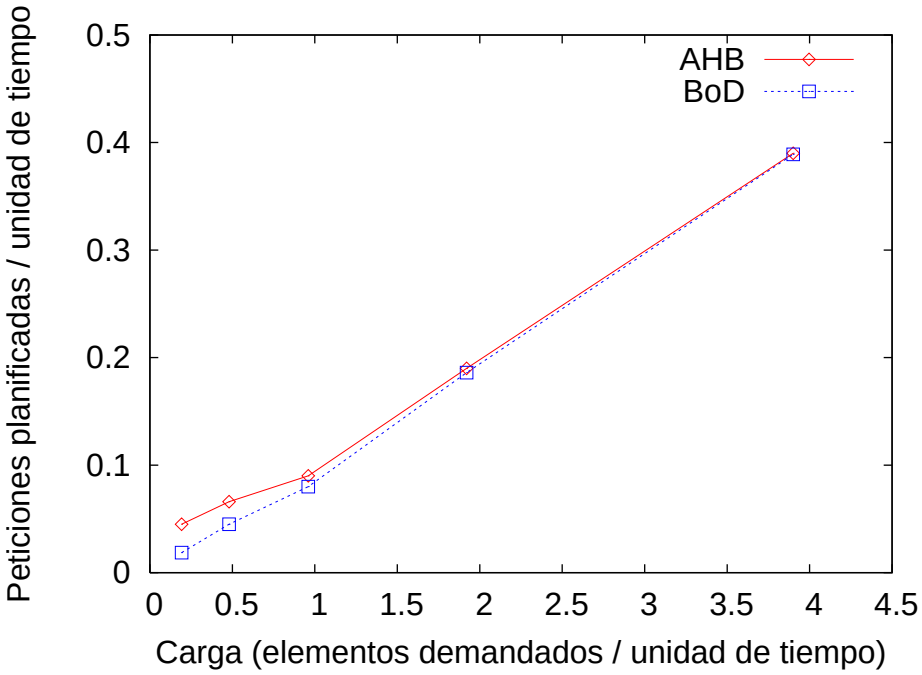

Figura 6.3. Distribución estática uniforme, peticiones planificadas

Aún así, no se produce saturación del canal de subida en ningún caso, ya que en el caso peor sólo deben ser planificadas 0.4 peticiones por unidad de tiempo. La razón es que el programa periódico se ajusta a las necesidades de los usuarios, los cuales no realizan peticiones para los elementos más populares, y el canal de subida no se congestiona.

6.2.2 Distribución dinámica de entrada

En la figura 6.4 podemos ver el experimento equivalente al anterior, utilizando ahora una versión dinámica de la distribución uniforme de dos niveles, en la cual los 20 elementos con mayor frecuencia de acceso cambian 5 veces durante la simulación. En concreto, inicialmente, los 20 elementos más demandados son [1,20], pasando a ser [201, 220], [401, 420], [601, 620], [801, 820] en los instantes de tiempo 20000, 40000, 60000, 80000.

La distribución dinámica utilizada implica que BoD sólo incluye en el programa periódico los elementos más demandados al inicio de la simulación (20% del tiempo), mientras que en el 80% del tiempo restante incluye elementos que no son los más demandados. Como consecuencia de ello, el número de plazos perdidos en BoD es mucho mayor que el de AHB en todo el rango de variación de la carga.

Debemos observar la similitud de las curvas de las figuras 6.4 y 6.2 para AHB, lo cual prueba la capacidad de adaptación del algoritmo a perfiles de usuario dinámicos. La diferencia de prestaciones (menor que un 10% en el peor caso) se debe al tiempo de adaptación del programa periódico cada vez que cambia la

distribución. Debemos pensar que en un caso más realista es improbable que se produzcan cambios tan drásticos en la distribución de acceso.

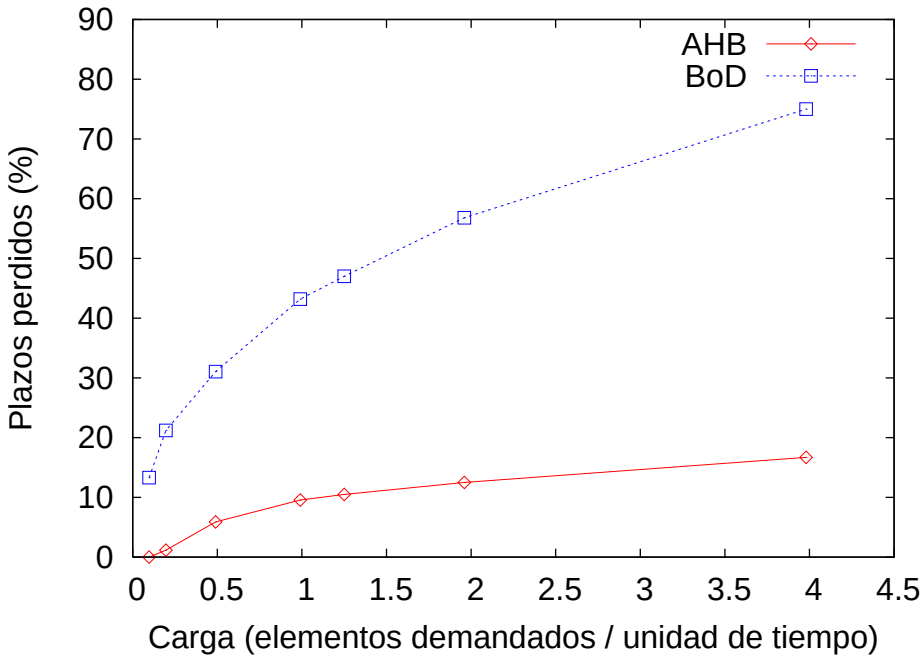

Figura 6.4. Distribución dinámica uniforme, plazos perdidos

En la figura 6.5 observamos que BoD ocasiona la saturación del canal de subida, no así AHB, que sólo necesita en el peor caso un 46% de la capacidad total (aumenta ligeramente con respecto a la distribución estática de entrada). La saturación del canal de subida en BoD es consecuencia directa de no incluir en el programa periódico los elementos más demandados, ya que esto provoca un fuerte incremento de peticiones en el canal de subida.

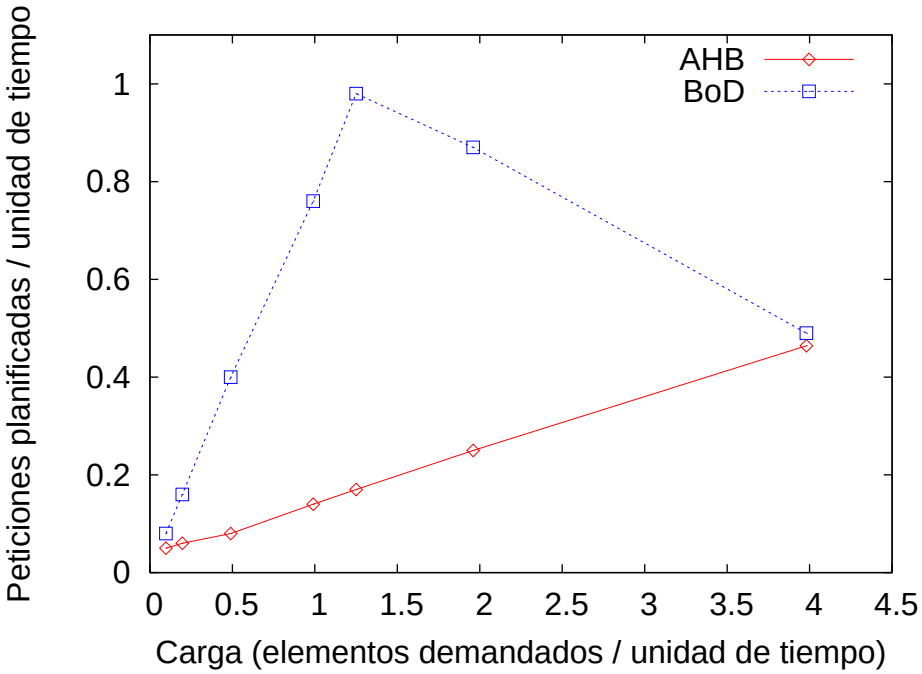

Figura 6.5. Distribución dinámica uniforme, peticiones planificadas

6.3 Comparación con modelos pull

En el siguiente conjunto de experimentos, vamos a comparar AHB con los siguientes modelos pull. Recordamos que los modelos basados en pull carecen de programa periódico, es decir, no están basados en ciclos.

- **EDF con agrupamiento**
  Se sirve el elemento con el menor plazo relativo. Se realiza agrupamiento si es posible, esto es, varias peticiones para un mismo elemento pueden ser

atendidas con una única transmisión si los plazos lo permiten.

- **MRF, Most Requested First**

  En este caso el elemento con mayor número de peticiones pendientes es servido en cada slot temporal [Wong88]. Esta política no considera la existencia de plazos asociados a las peticiones.

- **SIN-1**

  SIN-1 [Xu06], puede considerarse una combinación de EDF y MRF, es decir, tiene en cuenta tanto las peticiones pendientes como los plazos de cada elemento. En concreto, el elemento con el mínimo valor "sin-1" es transmitido en cada slot temporal, viniendo dado este valor por la siguiente fórmula:

$$SIN-1 = \frac{TimeToFirstDeadline_i}{PendingRequests_i}$$

La siguiente tabla muestra los parámetros más importantes de la simulación y los valores elegidos para ellos.

Tabla 6.2. Parámetros de simulación, modelos pull

Parámetro	Valor
Número de elementos	5000
Tamaño de cada elemento	10 páginas (=10 unidades de tiempo)
Plazos relativos	3000, 6000 o 9000 unidades de tiempo
Tiempo de simulación	1000000 unidades de tiempo
BW_Limit	0.9
Cooling_Factor	0.9
Expected_Sample_Size	5 peticiones
Capacidad del canal de subida	20 peticiones por unidad de tiempo
Rango de carga soportado	[1-64] elementos por unidad de tiempo

6.3.1 Canal de subida con capacidad infinita

Comenzamos considerando una capacidad del canal de subida ilimitada. Esta situación no es realista y favorece claramente a los modelos basados en pull, ya que no tienen en cuenta este factor. Para este conjunto de experimentos utilizaremos una distribución de entrada Zipftian.

Como observamos en la figura 6.6, las curvas de AHB y SIN-1 son bastante similares. En la región de la figura correspondiente a baja carga se comporta ligeramente mejor SIN-1, en torno a un 2% cuando la carga ofrecida son 10 elementos. Esto es debido a que el algoritmo de adaptación de AHB funciona peor cuando la carga ofrecida es baja, ya que el muestreo es más inexacto.

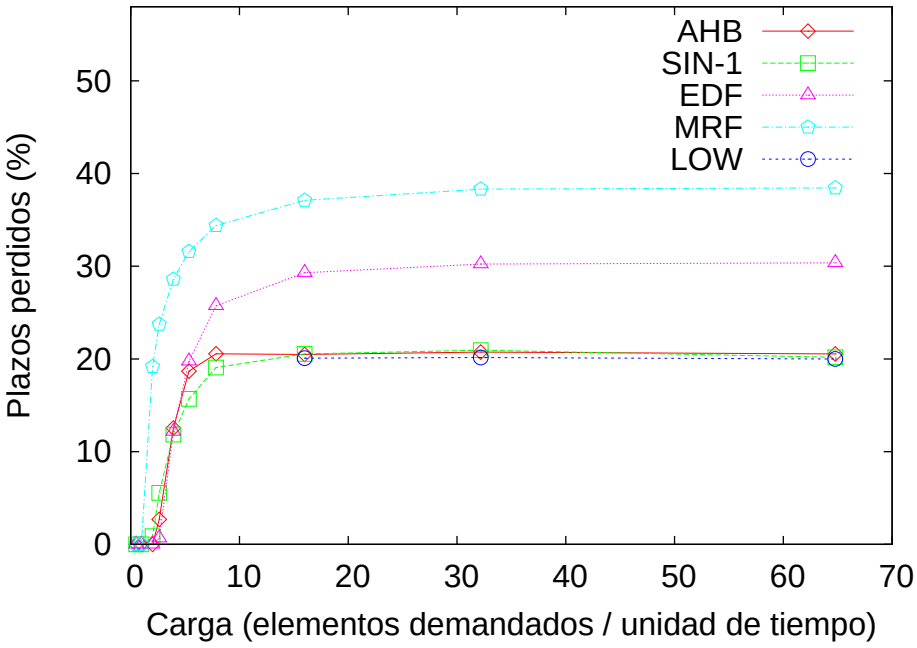

Figura 6.6. Canal de subida no limitado, plazos perdidos

MRF y EDF son claramente inferiores, debido principalmente a que MRF no tiene en cuenta los plazos temporales y EDF no tiene en cuenta las frecuencias de acceso. Como curiosidad podemos destacar que MRF se comporta de forma inferior a EDF, es decir, penaliza más no tener en cuenta los plazos de las peticiones que no tener en cuenta el número de las mismas.

En el capítulo 4 de esta tesis analizamos el límite inferior teórico del número de peticiones no satisfechas en un servidor de difusión, en condiciones de fuerte sobrecarga. Este límite ha sido incluido en la figura 6.6 como "LOW", y su valor es aproximadamente de un 20%. Como se muestra en la figura, tanto AHB con SIN-1 se encuentran muy próximos al límite teórico inferior, no así EDF ni MRF.

La figura 6.7 muestra el número de peticiones que deben ser planificadas. En los 3 modelos basados en pull es exactamente igual a la carga ofrecida, es decir, todas las peticiones deben ser planificadas por el servidor. Sin embargo, para AHB sólo es necesaria la planificación de aproximadamente un 30% de estas peticiones, lo cual implica un importante ahorro en recursos computacionales.

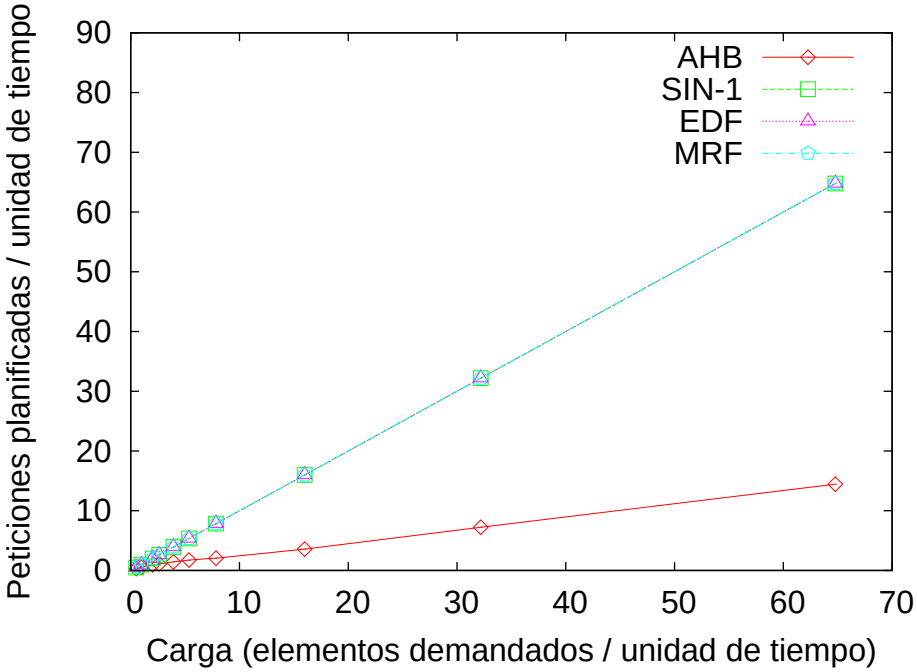

Figura 6.7. Canal de subida no limitado, peticiones planificadas

Es decir, incluso en el hipotético caso de que el ancho de banda del canal de subida fuera ilimitado, un sistema basado en SIN-1 (o cualquier otro modelo pull) obligaría a dimensionar un mayor número de recursos que AHB para hacer frente a una cantidad muy superior de peticiones a planificar por unidad de tiempo.

6.3.2 Canal de subida con capacidad finita

En el siguiente experimento el flujo máximo de peticiones es 64 elementos demandados por unidad de tiempo, lo cual sobrepasa en más de tres veces la capacidad máxima del canal (20 peticiones por unidad de tiempo). Por claridad en las figuras, sólo compararemos AHB con SIN-1, ya que las demás políticas de planificación son claramente inferiores a ambos modelos.

La figura 6.8 muestra que la limitación del canal de subida no tiene efecto alguno en AHB, ya que prácticamente no se produce saturación del canal, manteniéndose muy próximo al límite teórico inferior (LOW). La razón es que el programa periódico de AHB se adapta continuamente a las necesidades de los usuarios, evitando que éstos realicen un gran número de peticiones y manteniendo el tráfico del canal de subida en niveles razonables.

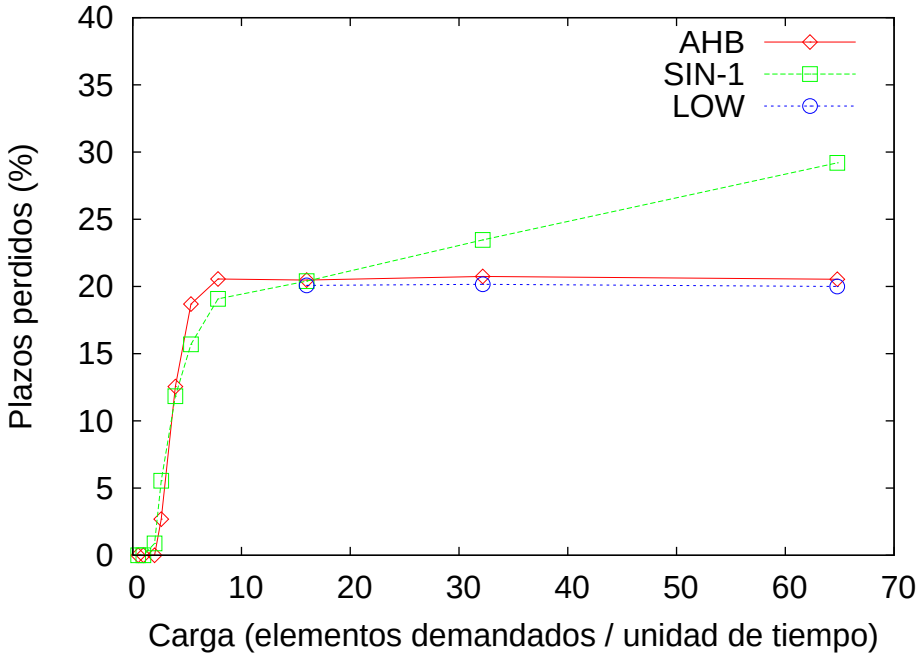

Figura 6.8. Canal de subida limitado, plazos perdidos

Sin embargo, como podemos ver en la figura 6.9, SIN-1 sufre el efecto de saturación del canal de subida. A partir del punto de saturación (cuando la carga ofrecida es mayor que 20) la peticiones de los usuarios empiezan a experimentar colisiones y no llegan al servidor. En consecuencia, el porcentaje de plazos perdidos aumenta, comportándose de forma inferior a AHB, aproximadamente un 50% peor en el peor caso (perdiendo alrededor de un 30% de los plazos frente a un 20% de AHB).

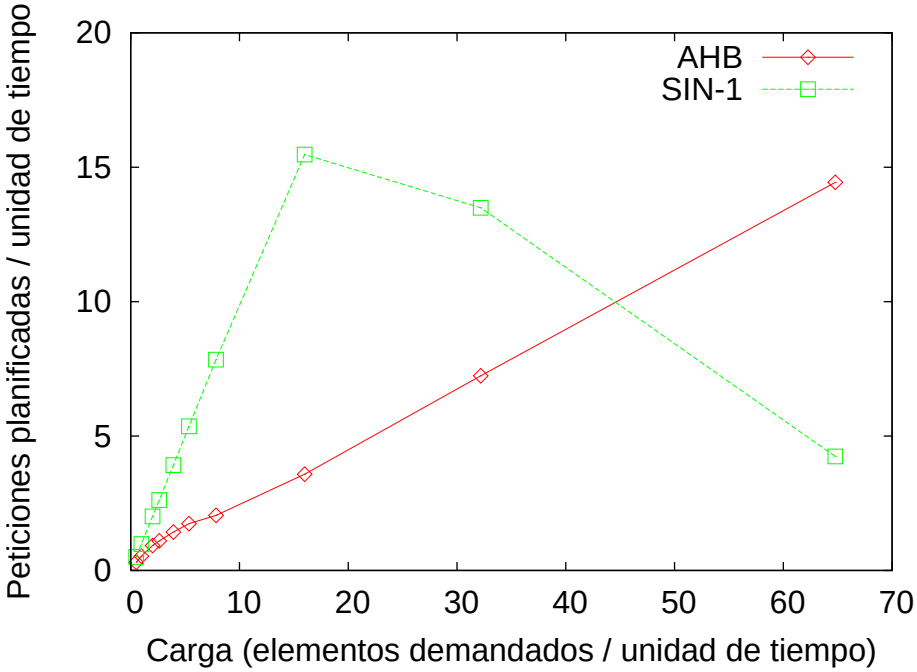

Figura 6.9. Canal de subida limitado, peticiones planificadas

6.4 Variación de parámetros en AHB

En los siguientes experimentos vamos a comprobar la influencia que tiene la variación de diferentes parámetros en el algoritmo AHB.

En la figura 6.10 podemos observar los diferentes valores de plazos perdidos en función de la variación del parámetro BW_Limit. Este parámetro limita el porcentaje máximo de ancho de banda que se puede asignar al programa periódico. Concretamente, hemos tomado 5 valores diferentes, entre 25% y 95%.

El resultado obtenido confirma que, para valores de carga bajos, BW_Limit no tiene ninguna influencia en el número de plazos perdidos. En cambio, para valores de carga altos, AHB se comporta mejor en tanto en cuanto el valor de BW_Limit sea mayor. Es lógico pensar que en condiciones de sobrecarga el algoritmo es más eficiente cuanto mayor sea el porcentaje destinado al programa periódico.

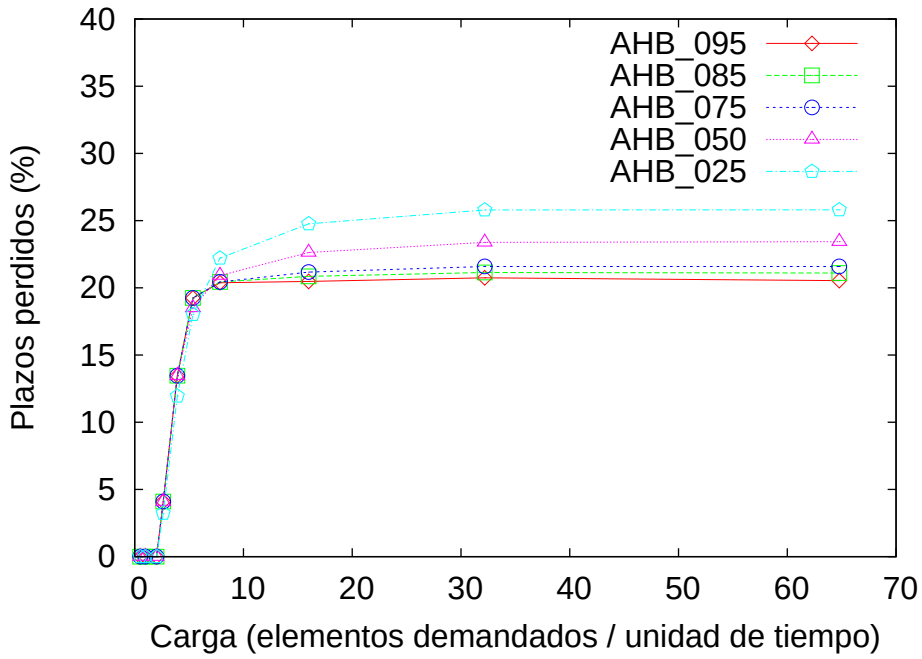

Figura 6.10. Variación del parámetro BW_Limit

La figura 6.11 representa la variación de plazos perdidos para diferentes valores del parámetro Cooling_Factor, entre 25% y 95%. Vemos que para la distribución de entrada utilizada, el valor más adecuado es el menor, es decir, 25%, sobre todo en condiciones de máxima sobrecarga (en condiciones de leve sobrecarga no existen diferencias apreciables). La explicación es que valores más bajos de Cooling_Factor implican una más rápida adaptación a perfiles dinámicos.

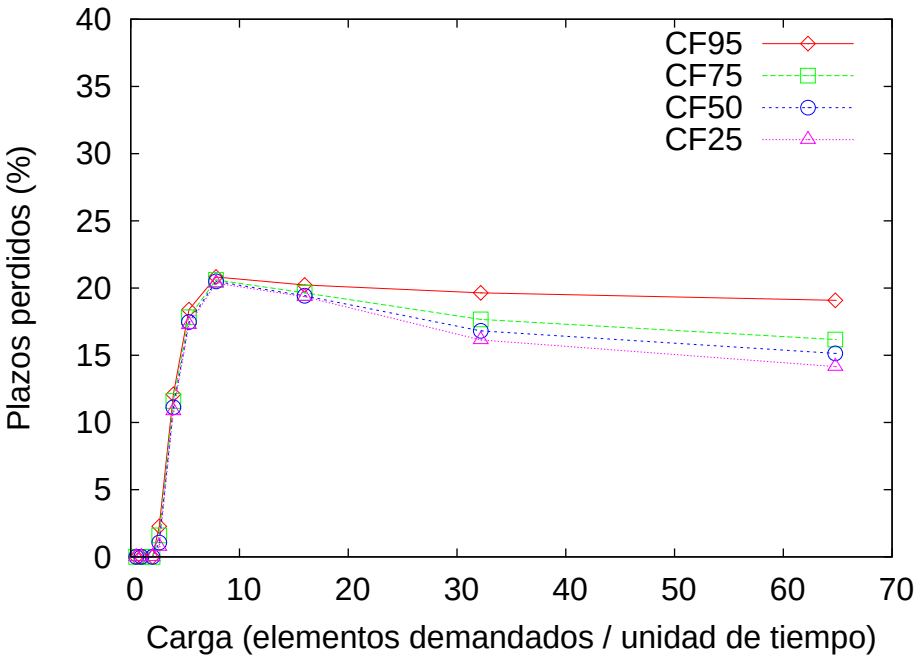

Figura 6.11. Variación del parámetro Cooling_Factor

En la figura 6.12 podemos apreciar las diferencias en cuanto a plazos perdidos cuando variamos el parámetro Expected_Sample_Size, concretamente entre 5 y 30. Vemos que se obtienen mejores resultados para valores más bajos de Expected_Sample_Size, ya que se produce una menor sobrecarga

en el sistema al disminuir el tiempo de muestreo para cada elemento en estado líquido.

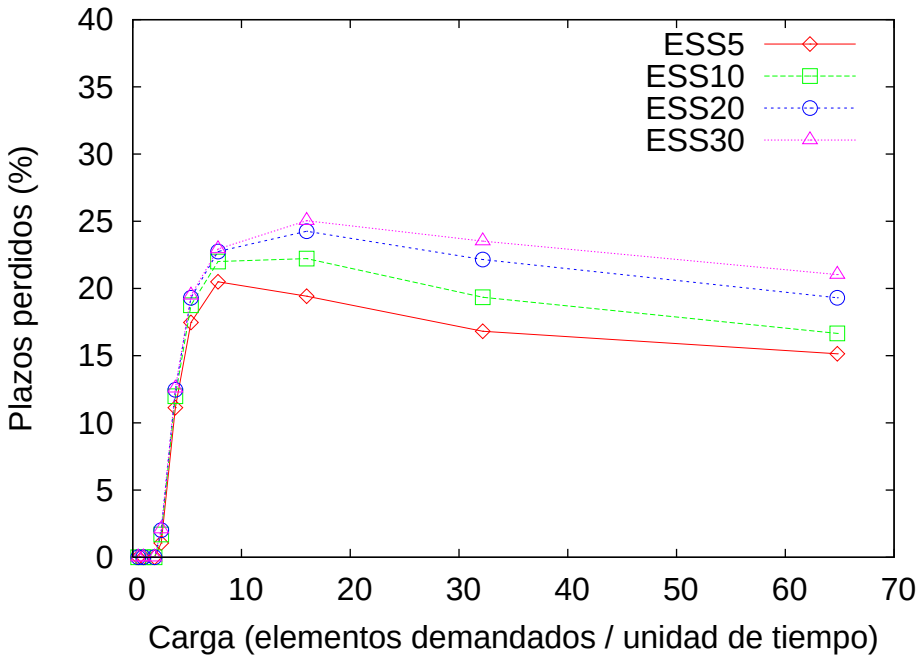

Figura 6.12. Variación del parámetro Expected_Sample_Size

6.5 Discusión de los resultados

De todos los modelos considerados, AHB es la única opción que obtiene unas prestaciones adecuadas en entornos de comunicación asimétricos con restricciones temporales, cuando la distribución de acceso es dinámica y el ancho de banda está limitado tanto en el canal de subida como en el de bajada.

Los modelos híbridos no adaptativos como BoD sólo funcionan satisfactoriamente cuando los perfiles de acceso son estáticos, e incluso en esta situación, es necesario que exista una diferencia

clara entre los elementos más demandados y menos demandados, para poder ajustar el programa periódico que es calculado a priori.

En cuanto a los modelos basados en pull, no son adecuados a no ser que se utilice un canal de subida de capacidad ilimitada (lo cual no es realista) y el servidor tenga recursos computacionales suficientes para planificar todas las peticiones enviadas por los usuarios.

Hemos visto experimentalmente que nuestro modelo AHB es capaz de adaptarse dinámicamente a diferentes patrones de acceso, evitando o reduciendo la saturación del canal de subida y satisfaciendo a un número mayor de usuarios que los modelos comparados. Esto se debe principalmente a la continua adaptación del contenido del programa periódico a las necesidades de los usuarios.

La figura 6.13 muestra la adaptación de la cantidad de ancho de banda del canal de bajada ocupada por el programa periódico en AHB para las diferentes distribuciones utilizadas.

Se puede apreciar que cuando la carga ofrecida es baja, sólo unos pocos elementos forman parte del programa periódico. En la distribución estática uniforme, vemos que el tamaño del programa aumenta hasta que los 20 elementos más demandados son incluidos en el programa, permaneciendo constante con el aumento de la carga.

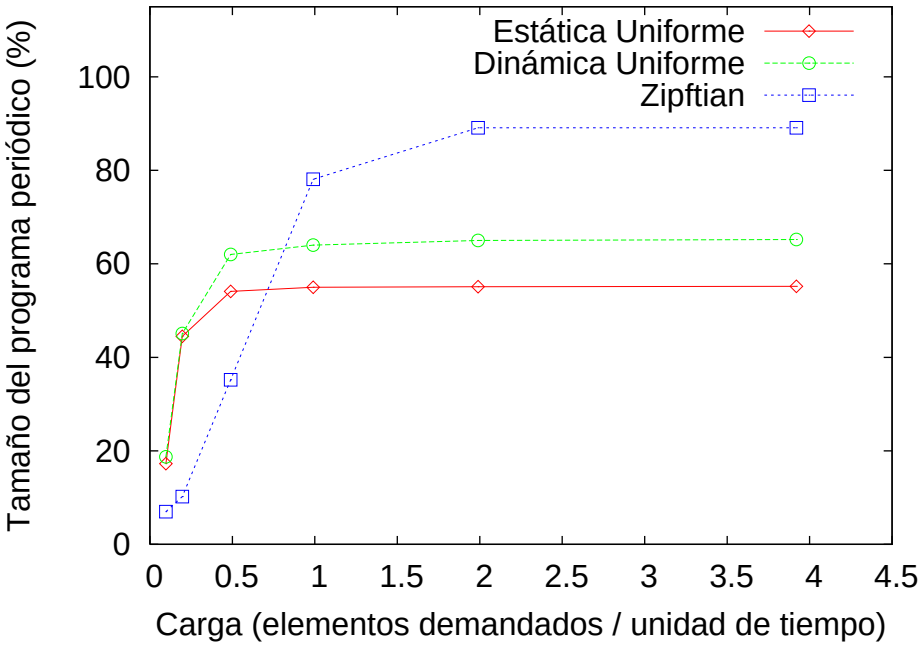

Figura 6.13. Ancho de banda relativo ocupado por el programa periódico en AHB

En el caso de la distribución dinámica uniforme, el programa periódico ocupa más ancho de banda debido al tiempo de permanencia de elementos en el programa periódico que ya no son populares durante varios ciclos, hasta que el factor de enfriamiento haga que su popularidad estimada descienda y sean extraídos del programa.

En la distribución Zipftian, el programa periódico va creciendo con el aumento de la carga hasta alcanzar BW_Limit.

Capítulo 7:

Conclusión

Con la creciente popularidad de aplicaciones para usuarios de dispositivos móviles, la transmisión utilizando difusión ha demostrado ser el modo más eficiente para diseminación de información a un gran número de usuarios con un alto grado de concordancia en sus demandas de información.

La necesidad de diseñar servidores de información eficientes y escalables adecuados a estos entornos es patente. En este trabajo de investigación hemos presentado, analizado y evaluado un nuevo modelo de servidor de información para entornos de comunicación asimétricos con restricciones temporales donde la población de usuarios presenta perfiles de acceso a la información dinámicos debido a la movilidad.

El modelo AHB presentado y analizado en la presente tesis utiliza difusión con modo híbrido de transmisión, esto es, transmisión periódica y bajo demanda, permitiendo a los usuarios demandar información con requisitos temporales.

Con el objetivo de maximizar el número de usuarios satisfechos, nuestra aproximación al problema trata de obtener los algoritmos de planificación que utilicen con máxima eficacia el ancho de banda disponible, combinando factores como frecuencias de acceso a la información, plazos temporales y consumo de ancho de banda.

Los resultados obtenidos mediante simulación ponen de manifiesto que utilizando un programa periódico que se adapte a las necesidades reales de los usuarios, es decir, transmitiendo periódicamente aquellos elementos con alta frecuencia de acceso y bajo consumo de ancho de banda, nuestro servidor de información es claramente superior en número de usuarios satisfechos y requisitos computacionales tanto a modelos híbridos no adaptativos como a modelos basados en pull.

Es especialmente relevante el hecho de que nuestro modelo tiene en cuenta que el ancho de banda en el canal de subida es limitado y que las frecuencias de acceso a los datos son cambiantes en un escenario realista (especialmente si los usuarios tienen movilidad).

El efecto de la limitación del ancho de banda en el canal de subida es muy importante. Nuestro modelo evita que un tráfico excesivo sature el canal de subida, pudiendo soportar una mayor carga efectiva que el resto de modelos evaluados.

El trabajo futuro incluye extender la funcionalidad de muestro modelo para incorporar mecanismos que permitan la difusión de información multimedia con calidades de servicio garantizadas.

También es interesante estudiar la posibilidad de incorporar técnicas de aprendizaje al servidor para variar algunos parámetros que hemos considerado fijos (*BW_Limit*, *Cooling_Factor*, *Expected_Sample_Size*), en función de la distribución de acceso a los datos, con el objetivo de mejorar las prestaciones.

Por último, otra línea de trabajo sería la modificación del algoritmo de planificación para considerar diferentes clases de usuarios, con diferentes prioridades en el acceso a la información, así como diferentes tipos de plazos (rígidos y no rígidos).

Publicaciones generadas

Artículos publicados en revistas con índice de impacto:

- Jesus Fernandez, Daniel Mozos, *"Efficient scheduling for mobile time-constrained environments"*, IET Electronics Letters Journal, Volume 43, Issue 22, 25 October 2007, pp. 1214-1215. Institution of Electrical Engineers, London, ISSN: 0013-5194. Indice de impacto 1.009 (JCR 2007).

- Jesus Fernandez, Krithi Ramamritham, *"Adaptive Dissemination of Data in Time- Critical Asymmetric Communication Environments"*, Mobile Networks and Applications Journal, Volume 9, Issue 5, Oct 2004, pp. 491 – 505. Springer Science+Business Media B.V. (formerly Kluwer Academic Publishers B.V.), ISSN 1383-469X. Indice de impacto 0.931 (JCR 2004).

Artículos publicados en conferencias internacionales:

- Jesus Fernandez, Daniel Mozos, *"Pull vs. Hybrid: Comparing Scheduling Algorithms for Asymmetric Time-Constrained Environments"*, The 2008 International Conference on Wireless Networks (ICWN'08), Las Vegas, Nevada, USA, July 14-17, 2008, pp. 222 - 228. Regular Research Paper, ICWN08 acceptance rate 29%.

- Jesus Fernandez, Daniel Mozos, *"Adaptive Hybrid Broadcast for Data Dissemination in Time-Constrained Asymmetric Communication Environments"*, 32nd IEEE Euromicro Conference on Software Engineering and Advanced Applications (SEAA), Cavtat/Dubrovnik (Croatia), August 28-September 1, 2006, pp. 438 -447.

- Jesus Fernandez, Krithi Ramamritham, *"Adaptive dissemination of data in timecritical asymmetric communication environments"*, Proceedings of the 11th Euromicro Conference on Real-Time Systems, York (UK), June 1999, pp. 195 - 203.

- Ping Xuan, Subhabrata Sen, Oscar Gonzalez, Jesus Fernandez, Krithi Ramamritham, *"Broadcast on Demand: Efficient and Timely Dissemination of Data in mobile Environments"*, Proceedings of the Third IEEE Real Time Technology and Applications Symposium Montreal (Canada), June 1997, pp. 38 - 48.

Referencias

[Acha95a] S. Acharya, R. Alonso, M. Franklin and S. Zdonik, “Broadcast Disks: Data Management for Asymmetric Communication Environments”, Proceedings of ACM SIGMOD International Conference, San Jose, CA, May 1995.

[Acha95b] S. Acharya, M. Franklin and S. Zdonik, “Dissemination-Based Data Delivery Using Broadcast Disks'”, IEEE Personal Communications, 2(6), December, 1995.

[Acha96] S. Acharya, M. Franklin and S. Zdonik, "Prefetching from a Broadcast Disk", Proceedings of 12th ACM International Conference on Data Engineering, New Orleans, LA, Feb 1996.

[Acha97] S. Acharya, M. Franklin and S. Zdonik, "Balancing Push and Pull for Data Broadcast", Proceedings of ACM SIGMOD International Conference, Tucson, AZ, May 1997.

[Acha98] S. Acharya and S. Muthukrishnan, "Scheduling ondemand broadcasts: New metrics and algorithms", Proceedings of the 4th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'98), pp. 43–54, Dallas, TX, USA, October 1998.

[Akso99] D. Aksoy and M. Franklin, "R x W: A scheduling approach for large-scale on-demand data broadcast", IEEE/ACM Transactions on Networking, vol. 7, no. 6, pp. 846-860, Dec. 1999.

[Amma85] M. Ammar and J. Wong, "The Design of Teletext Broadcast Cycles", Performance Evaluation vol. 5(4), pp. 235-242, 1985.

[Barb94] D. Barbara and T. Imielinski, "Sleepers and workaholics: Caching strategies for mobile environments", In Proceedings of ACM SIGMOD Conference on Management of Data, pp. 1-12, Minneapolis, MN, USA, May 1994.

[Baru94] S. Baruah, N. Cohen, C. Plaxton and D. Varvel, "Proportionate Progress: A Notion of Fairness in Resource Allocation", Algorithmica 15(6), pp. 600-625, 1996.

[Baru95] S. Baruah, "Fairness in periodic real-time scheduling", Proceedings of 16th Real-Time Systems Symposium, Pisa, Italy, December 1995.

[Baru97a] S. Baruah and A. Bestavros, "Pinwheel Scheduling for Fault-tolerant Broadcast Disks in Real-Time Database Systems", Proceedings of IEEE International Conference on Data Engineering, Birmingham, England, April 1997.

[Baru97b] S. Baruah and A. Bestavros, "Real-Time mutable broadcast disks", Proceedings of Second International Workshop on Real-Time Databases, Burlington, VT, September 1997.

[Baru97c] S. Baruah and S. Lin, "Improved scheduling of generalized pinwheel task systems", Proceedings of Fourth International Workshop on Real-Time Computer Systems Applications, Teipei, Taiwan, October 1997.

[Best96] A. Bestavros, "AIDA-based Real-Time Fault-Tolerant Broadcast Disks", Proceedings of Second IEEE Real-Time Technology and Applications Symposium, Boston, MA, June 1996.

[Bowe92] T. Bowen, et al. "The Datacycle Architecture", Communications of the ACM 35,(12), pp. 71-81, Dec. 1992.

[Bres99] L. Breslau, P. Cao, L. Fan, G. Phillips, S. Shenker, "Web Caching and Zipf-like Distributions: Evidence and Implications", Proceedings IEEE INFOCOM 99, pp. 126-134, 1999.

[Corm90] T. Cormen, C. Leiserson and R. Rivest, "Introduction to Algorithms", McGraw-Hill, 1990.

[Datt99] A. Datta, D. VanderMeer, A. Celik and V. Kumar, "Broadcast Protocols to Support Efficient Retrieval from Databases by Mobile Users", ACM Transactions on Database systems, 24(1), pp. 1-79, 1999.

[Fran96] M. Frankin and S. Zdonik, "Dissemination-Based Information Systems", IEEE Data Engineering Bulletin, 19(3), pp. 1-9, 1996.

[Hel99] A Helal, B Haskell, J L Carter, R Brice, D Woelk and M Rusinkiewicz, "Any Time Anywhere Computing: Mobile Computing Concepts and Technology", The Kluwer International Series in Engineering and Computer Science, Kluwer Academic Publishers, Boston/Dordrecht/London, 1999.

[Hu02] Hu, C.-L., & Chen, M.-S., "Dynamic data broadcasting with traffic awareness", Proceedings of 22nd IEEE International Conference on Distributed Computing Systems (ICDCS), pp. 112-119, 2002.

[Imie94a] T. Imielinski, S. Viswanathan and B. Badrinath, "Energy Efficient Indexing on Air", Proceedings of ACM SIGMOD Conference, May 1994.

[Imie94b] T. Imielinski, S. Viswanathan, "Adaptive Wireless Information Systems", Proceedings of SIGDBS Conference, Oct. 1994.

[Imie97] T. Imielinski, S. Viswanathan and B. Badrinath, "Data on Air: Organization and Access", IEEE Transactions on Knowledge and Data Engineering, Vol.9, No. 3, May/June 1997.

[Jian98] S. Jiang and N. H. Vaidya, "Response time in data broadcast systems: Mean, variance and trade-off", In Proceedings of International Workshop on Satellite-based Information Services (WOSBIS), October 1998

[Jian99] S. Jiang and N. H. Vaidya, "Scheduling data broadcasting to impatient users", In Proceedings of the ACM International Workshop on Data Engineering for Wireless and Mobile Access, pp. 52-59, Seattle WA USA, August 1999.

[Kara01] M. Karakaya and O. Ulusoy, "Evaluation of a Broadcast Scheduling Algorithm", Lecture Notes in Computer Science, volume 2151, pp. 182-195, 2001.

[Lee96] W.C. Lee and D. L. Lee, "Using signature techniques for information filtering in wireless and mobile

environments", Journal of Distributed and Parallel Databases (DPDB), 4(3): pp. 205-227, July 1996.

[Lee99] Lee, W.-C., Hu, Q., & Lee, D. L. , "A study on channel allocation for data dissemination in mobile computing environments", ACM/Kluwer Mobile Network and Applications, 4(2), pp. 117-129, 1999.

[Liu73] C.L. Liu and J.W. Layland, "Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environments", J. ACM, vol. 20, no. 1, pp. 46-61, 1973.

[Mok88] A. Mok, "Task Management Techniques for Enforcing ED scheduling on a Periodic Task Set", Proceedings of 5th IEEE Workshop on Real-Time Software and Operating Systems, Washington D.C., May 1988.

[Nico02] Nicopolitidis, P., Papadimitriou, G. I., & Pomportsis, A. S., "Using learning automata for adaptive push-based data broadcasting in asymmetric wireless environments", IEEE Transactions on Vehicular Technology, 51(6), pp. 1652-1660, 2002.

[Pino02] M. C. Pinotti and N. Saxena, "Push less and pull the current highest demanded data item to decrease the waiting time in asymmetric communication environments", 4th International Workshop on Distributed and Mobile Computing, (IWDC), Calcutta, India pp. 203-213 Springer-Verlag 2002; LNCS 2571, Dec. 2002.

[Saty01] M. Satyanarayanan, "Pervasive computing: Vision and challenges", *IEEE Personal Communications*, 8(4): pp. 10-17, Sept. 2001.

[Saut06] M. Sauter, "Communication Systems for the Mobile Information Society", John Wiley and Sons, Sept. 2006.

[Stat96] K. Stathatos, N. Roussopoulos and J. S. Baras, "Adaptive Data Broadcasting Using Air-Cache", Proceedings of the 1st International Workshop on Satellite-based Information Services, Rye, NY, 1996.

[Stat97] K. Stathatos, N. Roussopoulos and J. S. Baras, "Adaptive Data Broadcast in Hybrid Networks", Proceedings of the 23rd VLDB Conference, Athens, Greece, 1997.

[Tann96] A. Tanenbaum, "Computer Networks, Third Edition", Prentice-Hall, 1996.

[Vaid99] N. H. Vaidya and S. Hameed, "Scheduling data broadcast in asymmetric communication environments", Wireless Networks, 5: pp. 171-182, 1999.

[Weis91] M. Weiser, "The Computer for the 21st Century", *Scientific American, 1991*.

[Wong88] J. W. Wong, "Broadcast delivery", Proc. IEEE, vol. 76, no. 12, pp. 1566-1577, Dec. 1988.

[Xu06] J. Xu, X. Tang and W. C. Lee, "Time-Critical On-Demand Broadcast: Algorithms, Analysis and Performance

Evaluation", IEEE Transactions on Parallel and Distributed Systems, vol. 17, no. 1, pp. 3-14, Jan. 2006.

[Xuan97] P. Xuan, S. Sen, O. Gonzalez, J. Fernandez and K. Ramamritham, "Efficient and Timely Dissemination of Data in mobile Environments", Proceedings of the Third IEEE Real Time Technology and Applications Symposium, Montreal, Canada, June 1997.

[Zip49] G. K. Zipf, "Human Behaviour and the Principles of Least Effort", Adison-Wesley, Reading MA, 1949.