El rendimiento de los protocolos de replicación de consistencia débil

Richard A. Golding
Darrell D. E. Long

UCSC-CRL-92-30 6

de julio de 1992

Laboratorio de Sistemas
Concurrentes Ciencias de la
Computación y la Información
Universidad de California, Santa
Cruz Santa Cruz, CA 95064

Los protocolos de replicación de consistencia débil pueden utilizarse para construir servicios de área amplia que sean escalables, tolerantes a fallos y útiles para sistemas informáticos móviles. Hemos desarrollado el protocolo *anti-entropía con marca de tiempo*, que proporciona una entrega eventual fiable con una variedad de ordenamientos de mensajes. Los pares de réplicas intercambian periódicamente mensajes de actualización; de este modo, las actualizaciones se propagan finalmente a todas las réplicas. En este trabajo presentamos un análisis detallado de la tolerancia a fallos y la consistencia que proporciona este protocolo. El protocolo es extremadamente robusto frente a los fallos del sitio y de la red, y se adapta bien a un gran número de réplicas.

1 Introducción

Estamos investigando una arquitectura para construir servicios distribuidos que haga hincapié en la escalabilidad y la tolerancia a los fallos. Esto permite que las aplicaciones respondan con elegancia a los cambios en la demanda y a los fallos del sitio y de la red. También proporciona un mecanismo único para soportar servicios de área amplia y sistemas informáticos móviles. Utiliza técnicas de replicación de consistencia débil para construir un servicio distribuido flexible.

Utilizamos *la replicación de* datos para satisfacer las demandas de disponibilidad y permitir la escalabilidad. La replicación es *dinámica, ya* que pueden añadirse o eliminarse nuevos servidores para adaptarse a los cambios en la demanda. El sistema es *asíncrono* y los servidores son lo más *independientes posible*; nunca requiere la cooperación sincrónica de un gran número de sitios. Esto mejora su capacidad para manejar tanto la comunicación como los fallos de los sitios.

Los protocolos de replicación *eventual* o *débilmente consistente* no realizan actualizaciones sincrónicas. En su lugar, las actualizaciones se entregan primero a un sitio y luego se propagan de forma asíncrona a los demás. El valor que un servidor devuelve a una solicitud de lectura de un cliente depende de si ese servidor ya ha observado la actualización. Al final, todos los servidores observarán la actualización. Varios sistemas de información existentes, como Usenet [1] y el sistema Xerox Grapevine [2], utilizan técnicas similares.

La propagación retardada significa que los clientes no esperan a que las actualizaciones lleguen a sitios distantes, y la tolerancia a fallos de los datos replicados no puede verse comprometida por clientes que se comporten mal. También permite que las actualizaciones se envíen mediante protocolos de transferencia masiva, que proporcionan la mejor eficiencia en redes de gran ancho de banda y alta latencia. Estas transferencias pueden producirse en horas de poca actividad. Las réplicas pueden desconectarse de la red durante un periodo de tiempo, y se actualizarán una vez que se vuelvan a conectar. Por otro lado, los clientes deben ser capaces de tolerar cierta inconsistencia, y la aplicación puede necesitar proporcionar un mecanismo para reconciliar las actualizaciones conflictivas.

Un gran número de réplicas permite colocarlas cerca de los clientes y repartir la carga de las consultas entre más sitios. Esto disminuye tanto la latencia de la comunicación para las solicitudes de los clientes como la cantidad de tráfico de larga distancia que debe transportarse en los enlaces de la red troncal. Los sistemas informáticos móviles pueden mantener una réplica local, garantizando que los usuarios puedan utilizar la información de acceso incluso cuando están desconectados de la red.

Estos protocolos pueden compararse con los protocolos de replicación consistente, como los protocolos de votación. Los protocolos consistentes no pueden manejar prácticamente cientos o miles de réplicas, mientras que los protocolos de consistencia débil sí. Los protocolos consistentes requieren la participación sincrónica de un gran número de réplicas, lo que puede ser imposible cuando un cliente reside en un sistema portátil o cuando la red está dividida. También es difícil compartir la carga de procesamiento entre muchas réplicas. El tráfico de comunicaciones y la latencia asociada suelen ser inaceptablemente grandes para un servicio con réplicas dispersas en varios continentes.

1.1 El entorno de Internet

Internet tiene varios comportamientos que deben tenerse en cuenta a la hora de diseñar un servicio distribuido de área amplia. Entre ellos están la latencia necesaria para enviar mensajes, que puede afectar al tiempo de respuesta de una aplicación, y la falta de fiabilidad de las comunicaciones, que puede requerir protocolos de comunicación robustos. Por ejemplo, dos hosts en una Ethernet pueden intercambiar un par de datagramas en unos pocos milisegundos, mientras que dos hosts en el mismo continente pueden necesitar entre 50 y 200 milisegundos. Los hosts de distintos continentes pueden requerir incluso más tiempo. Las tasas de pérdida de paquetes del 40% son comunes, y pueden ser mucho más altas [3]. Internet tiene muchos puntos únicos de fallo, y en un momento dado suele estar dividida en

varias redes no comunicadas. Este es un entorno difícil para construir aplicaciones distribuidas.

La aplicación también debe manejar el gran número de usuarios que pueden acceder a un servicio ampliamente disponible. En la actualidad, Internet cuenta con más de 900.000 hosts [4]; la base de usuarios potenciales se cuenta por millones, y se espera que estas cifras aumenten rápidamente. El servicio anónimo de localización FTP **Archie** informó de que se realizaban del orden de 10.000 consultas al día (0,12 consultas por segundo) utilizando dos servidores en noviembre de 1991 [5]. Los servicios utilizados por un público más amplio observarían una carga varios órdenes de magnitud mayor. Esperamos encontrar servicios en

en un futuro próximo procesar varios cientos de consultas por segundo, una carga mayor de la que pueden manejar los sistemas informáticos individuales o los enlaces de red individuales.

A pesar de este entorno, los usuarios esperan que un servicio se comporte como si se prestara en un sistema local. El tiempo de respuesta de una aplicación de área amplia no debe ser mucho mayor que el de una local. Además, los usuarios esperan utilizar el servicio mientras sus sistemas locales funcionen. Se trata de una expectativa especialmente difícil de cumplir en los sistemas portátiles, donde el sistema puede estar desconectado de la red durante mucho tiempo o puede estar "semiconectado" mediante una costosa conexión de bajo ancho de banda. Varios investigadores están estudiando sistemas de archivos que puedan tolerar la desconexión [6, 7].

Suponemos que los procesos del servidor tienen acceso a un almacenamiento pseudoestable, como un disco magnético, que no se verá afectado por una caída del sistema. Los sitios también tienen relojes poco sincronizados. Los sitios y los procesos fallan por colapso; es decir, cuando fallan no envían mensajes inválidos a otros procesos y no corrompen el almacenamiento estable. Los procesos pueden fallar temporalmente y luego recuperarse. Los sitios tienen dos modos de fallo: fallos temporales recuperables y retirada permanente del servicio. La red es lo suficientemente fiable como para que dos procesos cualesquiera puedan eventualmente intercambiar mensajes, pero nunca tiene que estar libre de particiones. Son posibles las *semiparticiones*, cuando sólo se dispone de una conexión de bajo ancho de banda.

2 Descripción del protocolo

Los datos replicados pueden implementarse como un grupo de procesos de réplica que se comunican a través de un *protocolo de comunicación de grupo*. El protocolo de comunicación de grupo generalmente proporciona un servicio de *multidifusión* que envía un mensaje de un proceso a todos los demás procesos del grupo. El protocolo también determina la *consistencia* de cada réplica controlando el orden de envío de los mensajes entre los procesos.

Los protocolos de consistencia débil garantizan que los mensajes se entregan a todos los miembros, pero no garantizan cuándo. En esta sección discutimos cómo la consistencia débil se compara con otros tipos de consistencia, y resumimos un protocolo de replicación de consistencia débil.

La garantía de entrega fiable no se cumple en un caso importante: cuando un proceso falla permanentemente y pierde datos. Ningún esquema de comunicación de consistencia débil puede librarse de esto, ya que existe una ventana de vulnerabilidad mientras los datos se envían a otros procesos. En la práctica, la duración es insignificante. En la sección 3 presentamos un análisis de la pérdida de información.

2.1 Tipos de consistencia

En general, dos procesos son coherentes en el momento t si han recibido el mismo conjunto de mensajes. Los diferentes grados de consistencia imponen diferentes restricciones a los órdenes en los que se pueden entregar los mensajes. A diferencia de otros trabajos sobre consistencia distribuida, siempre medimos la consistencia en tiempo *real*, en lugar de utilizar una medida de tiempo virtual.

El servicio prestado por un proceso depende de los mensajes que ha recibido. Si una réplica debe proporcionar un cierto nivel de consistencia, debe utilizar un protocolo de comunicación que garantice un nivel de consistencia similar. Los protocolos de comunicación pueden ofrecer garantías sobre la *entrega de los mensajes, el orden de entrega* y el *tiempo de entrega.* En general, las garantías fuertes requieren protocolos síncronos multifase, mientras que las garantías más débiles permiten protocolos asíncronos eficientes.

Los mensajes pueden entregarse *de forma fiable, en cuyo* caso la llegada está garantizada, o con *el mejor esfuerzo,* lo que significa que el sistema hará un intento de entregar el mensaje, pero la entrega no está garantizada.

Los mensajes se entregarán a los procesos en algún orden, tal vez diferente del orden en que se reciben. Hay varios órdenes posibles, entre ellos el total, el causal, el de inconsistencia de límites, el del canal FIFO y el de recepción. Un orden *total* significa que todos los procesos verán los mismos mensajes en el mismo orden, aunque ese orden no será necesariamente el orden en que se enviaron los mensajes. El ordenamiento *causal implica* que cualquier mensaje con una potencial relación causal será entregado en el mismo orden en todas las réplicas [8, 9]. Los mensajes sin relación causal pueden ser entregados en diferentes órdenes en diferentes procesos. Un ordenamiento de *inconsistencia limitada* asegura que la base de datos en un sitio nunca difiere del valor global correcto por más de una constante [10, 11]. Las ordenaciones más débiles incluyen una ordenación *por proceso* o por *canal FIFO*, donde los mensajes de cualquier proceso particular se entregan en orden, pero los flujos de mensajes de diferentes procesos pueden intercalarse arbitrariamente. Por último, existe la posibilidad de entregar simplemente los mensajes en el orden en que se reciben, sin tener en cuenta el orden.

El protocolo de comunicación puede entregar los mensajes *de forma sincrónica,* en un tiempo *limitado*, o *eventualmente*
en un tiempo finito pero ilimitado.

Los requisitos de consistencia fuerte son imposibles de cumplir en los casos más generales. Por ejemplo, si no hay límites en el tiempo de entrega de los mensajes, no es posible garantizar la consistencia [12]. Además, si los procesos pueden fallar de forma arbitraria, proporcionar una entrega fiable equivale a un acuerdo bizantino. Para la mayoría de las aplicaciones, Internet puede tratarse como una red no fiable,

limitada y de difusión (en contraposición a la red estricta punto a punto).

Los protocolos de consistencia débil que hemos desarrollado proporcionan una entrega fiable, y pueden modificarse para producir varios ordenamientos de entrega, pero sólo garantizan la entrega eventual de mensajes. En particular, existe una probabilidad no nula de que dos procesos hayan recibido todos los mismos mensajes, y todos los procesos son

Se garantiza que se pondrán de acuerdo en un tiempo finito pero ilimitado si no se envían más mensajes.

Grapevine [2] fue uno de los primeros sistemas de área amplia en utilizar la consistencia débil. En ese sistema, los datos replicados se actualizaban primero en un sitio, y luego los resultados se propagaban a otros sitios en segundo plano. Las actualizaciones se propagaban de tres maneras. Un sitio puede utilizar primero el *correo directo,* una multidifusión poco fiable, para hacer llegar la actualización al mayor número de sitios posible. A continuación, utilizaba el *rumor* para propagar las actualizaciones recientes de un sitio a otro. Por último, los pares de sitios intercambiarían periódicamente todas las actualizaciones conocidas en una *sesión de antientropía* hasta que fueran mutuamente coherentes. De los tres métodos, sólo la antientropía garantizaba la entrega a todos los sitios.

2.2 Antientropía con sello de tiempo

Hemos desarrollado un nuevo protocolo de comunicación de grupo que proporciona una entrega fiable y eventual, llamado *antientropía con marca de tiempo* [13]. Dado que el protocolo es tolerante a fallos, los mensajes se entregarán a todos los procesos del grupo incluso si los procesos fallan temporalmente o se desconectan de la red. También hemos desarrollado un mecanismo de pertenencia a grupos relacionado que se encarga de añadir y eliminar procesos del grupo de réplicas [14]. Las desventajas son que el protocolo puede tener que retrasar la entrega de mensajes (es un protocolo de bloqueo), que las réplicas deben mantener registros en el disco que no se vean comprometidos por los fallos y la recuperación, y que la información de la marca de tiempo debe añadirse a cada mensaje.

Cada réplica mantiene tres estructuras de datos: un *registro de mensajes* y dos *vectores de marcas de tiempo.* Todos ellos deben mantenerse en un almacenamiento estable, para que no se corrompan cuando el sitio o el proceso se bloqueen. El almacenamiento estable es necesario para cumplir estrictamente con las garantías de fiabilidad; sin embargo, hemos comprobado que un cuidadoso almacenamiento en búfer en un almacenamiento volátil no compromete apreciablemente esta garantía. Cada sitio también debe mantener un reloj que esté vagamente sincronizado con otros sitios. [1]

El registro de mensajes contiene los mensajes que ha recibido un proceso. Los mensajes se sellan con la identidad de la réplica que inició el mensaje y una marca de tiempo. Los mensajes se introducen en el registro cuando se reciben y se eliminan cuando todas las demás réplicas los han recibido.

Las réplicas mantienen un vector de marcas de tiempo de *resumen* para registrar las actualizaciones que han recibido. El vector de marcas de tiempo contiene una marca de tiempo para cada réplica en el grupo. La réplica *A* registra una marca de tiempo *t* para la réplica *B* cuando la réplica *A* ha recibido todos los mensajes de actualización enviados desde *B hasta* el momento *t.* El vector proporciona un mecanismo rápido para transmitir información resumida sobre el estado de una réplica.

Cada réplica también mantiene un *vector de marcas de tiempo de reconocimiento* para registrar qué mensajes han sido reconocidos por otras réplicas. Una réplica puede determinar que todas las demás réplicas han observado un mensaje en particular mirando sólo su vector de acuse de recibo local. Si la réplica *A* tiene una marca de tiempo *t* para la réplica *B, la* réplica *A* sabe que *B* ha recibido todos los mensajes de cualquier remitente con una marca de tiempo menor o igual a *t.* El proceso *B* establece periódicamente su entrada en su vector de acuse de recibo a la marca de tiempo mínima registrada en su vector de resumen. Este mecanismo avanza siempre que los relojes del sitio estén poco sincronizados y el vector de acuse de recibo se actualice regularmente.

Una réplica puede utilizar el vector de acuse de recibo para detectar cuándo todas las demás réplicas han recibido una actualización. En particular, para un mensaje de actualización enviado en el tiempo (real) *t*, hay un conjunto *M* de procesos de réplica en *t*. Un proceso de réplica *p* puede determinar eventualmente que todas las réplicas en *M* han recibido el mensaje o han fallado comparando la marca de tiempo del mensaje con las marcas de tiempo en su vector de acuse de recibo.

De vez en cuando, un proceso *A* seleccionará un proceso asociado *B* e iniciará una sesión *antientropía*. Una sesión comienza con los dos procesos asignando una marca de tiempo de sesión, y luego intercambiando sus vectores de resumen y acuse de recibo. Cada proceso determina si tiene mensajes que el otro quizás aún no ha observado, detectando que algunas de sus marcas de tiempo de resumen son mayores que las correspondientes de su compañero. Estos mensajes se recuperan del registro y se envían al otro proceso mediante un protocolo de flujo fiable.

1También hemos desarrollado un protocolo similar que requiere *O(n2)* estado por proceso en lugar de *O(n),* pero permite relojes no sincronizados. Este protocolo alternativo fue descubierto independientemente por Agrawal y Malpani [15].

La sesión termina con un intercambio de mensajes de acuse de recibo. Si cualquier paso del intercambio falla, cualquiera de los dos procesos puede abortar la sesión.

Al final de una sesión exitosa, ambos procesos han recibido el mismo conjunto de mensajes. Los procesos *A* y *B* fijan sus vectores de resumen y acuse de recibo al máximo elemental de su vector actual y el recibido del otro proceso.

Una vez finalizadas las sesiones de antientropía, se pueden entregar mensajes de actualización del registro a la base de datos y se pueden purgar las entradas de registro innecesarias. Una entrada de registro puede ser purgada cuando todos los demás procesos la han observado, lo cual es cierto cuando la marca de tiempo mínima en el vector de reconocimiento es mayor que la marca de tiempo en la entrada de registro.

Aquí el protocolo puede producir diferentes ordenamientos de entrega de mensajes. Si el sistema garantiza un ordenamiento total o causal, entonces sólo se podrán entregar aquellos mensajes con una marca de tiempo menor que la marca de tiempo mínima en el vector resumen. Los demás mensajes se retrasarán hasta que lleguen mensajes de otras réplicas. Si se permite un ordenamiento por proceso o más débil, los mensajes pueden ser entregados inmediatamente después de su recepción.

2.3 Variaciones

Existen diversas variaciones del protocolo básico de antientropía con sello de tiempo. Los vectores de marca de tiempo y las marcas de tiempo de los mensajes pueden utilizarse para ordenar los mensajes antes de que se entreguen desde el registro. Las sesiones anti-entropía pueden ser aumentadas por una multidifusión de mejor esfuerzo para acelerar la propagación, y las réplicas pueden utilizar diferentes políticas para seleccionar a los socios.

Como se ha mencionado anteriormente, son posibles muchos órdenes de entrega de mensajes diferentes. Un orden total es el más fuerte, y asegura que todas las réplicas procesarán todas las actualizaciones en el mismo orden. Esto puede hacerse entregando los mensajes en el orden de sus marcas de tiempo. Cuando los relojes del sitio siguen la condición de Lamport de "sucede antes" [8], este orden también respetará la causalidad. Un mensaje puede ser entregado en una réplica cuando el vector resumen no tiene marcas de tiempo menores que la marca de tiempo del mensaje. El ordenamiento del canal FIFO resulta cuando los mensajes de cada réplica se ordenan por marca de tiempo. Los mensajes pueden entonces entregarse antes que con los órdenes totales o causales. El retraso necesario para ordenar correctamente los mensajes depende de la rapidez con la que se propagan los mensajes entre las réplicas.

La multidifusión de mejor esfuerzo puede combinarse con la antientropía para difundir rápidamente la información. Cuando una réplica origina un mensaje, puede realizar una multidifusión a otras réplicas. Algunas de ellas no recibirán la multidifusión, ya sea porque la red dejó caer el mensaje o porque la réplica no estaba disponible temporalmente. Estos sitios recibirán el mensaje más tarde, cuando realicen una sesión de antientropía con otro sitio que haya recibido el mensaje. Esto puede acelerar la difusión cuando el orden de entrega del mensaje no es importante.

Hay varias políticas diferentes que los procesos de réplica pueden seguir a la hora de seleccionar socios anti-entropía. La selección *aleatoria* es la más sencilla. Sin embargo, hay otras opciones que pueden mejorar la velocidad de propagación de las actualizaciones o minimizar la comunicación a larga distancia. La selección del socio *más antiguo* selecciona siempre el socio del que la réplica no ha recibido una actualización en el mayor tiempo, con la esperanza de que este sitio tenga la mayor cantidad de actualizaciones que la réplica aún no ha observado. La selección de pareja *con sesgo de distancia* intenta evitar la comunicación a larga distancia en la medida de lo posible, ponderando las posibilidades de selección aleatoria de una pareja en función de su distancia. El sesgo de distancia se estudió como una optimización para Grapevine [16]. Los efectos de rendimiento de las diferentes selecciones de políticas se presentan en la Sección 4.

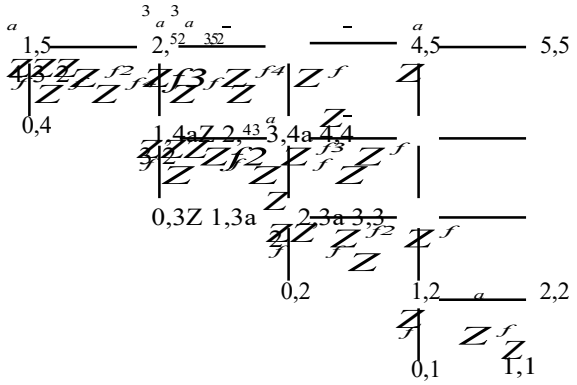

FIGURA 1: Modelo de fallo para cinco procesos. Este modelo sólo incluye el fallo permanente; el fallo temporal y la recuperación añadirían una dimensión adicional de estados.

3 Tolerancia a los fallos

En primer lugar, consideramos la frecuencia con la que los fallos del sitio impedirían que un protocolo de comunicación de consistencia débil entregara un mensaje a todas las réplicas.

3.1 Modelización analítica

Modelamos la pérdida de información mediante un sistema de transición de estados como el que se muestra en la figura 1. Cada estado se etiqueta con un par *hm, f i*, donde *m* es el número de réplicas en funcionamiento que han observado un mensaje, y *f* es el número total de réplicas en funcionamiento. El sistema comienza con una réplica que ha observado un mensaje de las *n* posibles (5 en el ejemplo). A continuación, el sistema puede propagar la información utilizando la antientropía, o una réplica puede ser retirada del servicio.

Tratamos la antientropía como un proceso de Poisson con tasa *J\a*. La tasa de sesiones *útiles de* anti-entropía, donde una réplica que tiene una actualización se pone en contacto con otra que no la tiene, es una función de *m* y *f* y de la política de selección de socios. En particular, *f* réplicas iniciarán sesiones de anti-entropía. Si las réplicas eligen a sus socios aleatoriamente, cada réplica que ha observado la actualización tiene una probabilidad *(f _ m)/(f _ 1)* de contactar con una réplica que aún no ha observado la actualización. Como la antientropía es un proceso de Poisson, la tasa de sesiones útiles de antientropía es

$$f\frac{f-m}{f-1}J\backslash a.$$

Para esta evaluación, tratamos la retirada del servicio como un proceso de Poisson con la tasa *J|f*.

Como la retirada del servicio es un evento permanente, el gráfico de transición de estados es acíclico, con *O(n2)* estados en el número de réplicas. La probabilidad $_{pi}$ de alcanzar cada estado *i* puede calcularse mediante un recorrido secuencial de los estados. Las funciones de densidad de probabilidad $_{pi}(t)$ del momento en que el sistema entra en cada estado pueden obtenerse de forma analítica o numérica. La solución analítica para $_{pi}(t)$ puede encontrarse convolucionando la distribución del tiempo de entrada *pj(t)* para cada estado predecesor *j* con la densidad de probabilidad del tiempo requerido para la transición de *j* a *i*. Obtuvimos una solución numérica utilizando una simple evaluación de Monte Carlo.

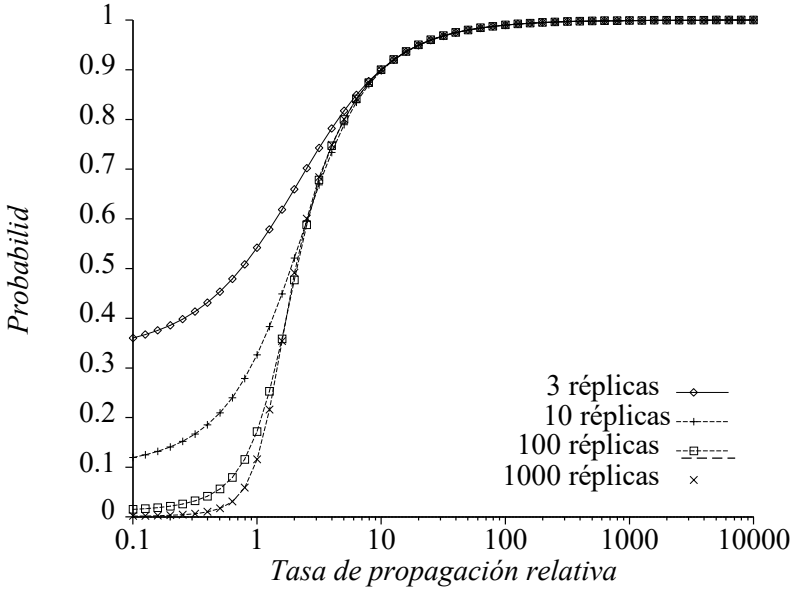

FIGURA 2: Probabilidad de entregar con éxito un mensaje a todos los sitios. La tasa de propagación relativa p es la relación entre la tasa de antientropía y la tasa de fallo de los sitios permanentes.

3.2 Resultados

La figura 2 muestra la probabilidad de éxito de la entrega para diferentes números de réplicas, teniendo en cuenta sólo los fallos permanentes. La probabilidad es una función de $p = J\backslash a/J\backslash f$, la relación entre la tasa de antientropía y la tasa de fallos permanentes del sitio. Los emplazamientos suelen ser retirados del servicio tras varios años de servicio, y las retiradas no programadas son poco frecuentes. Es probable que la tasa de antientropía sea muchos miles de veces mayor que la tasa de fallos permanentes. Como resultado, esperamos que no se pierdan mensajes por la retirada del servicio en un sistema típico.

Algunas implementaciones pueden almacenar mensajes en el almacenamiento volátil antes de copiarlos en el almacenamiento estable. Este es el comportamiento por defecto de varios sistemas operativos, incluyendo Unix. Estas implementaciones perderán la información en el almacenamiento volátil cuando una réplica falle temporalmente y se recupere.

El almacenamiento volátil complica el modelo. Los estados deben etiquetarse con cuatro valores: el número de réplicas en funcionamiento que no han observado un mensaje, el número que lo ha escrito en el almacén volátil, el número que lo ha escrito en el disco y el número que ha fallado temporalmente. Las transiciones de estado son complejas y la solución es poco práctica para un número realista de réplicas. Realizamos una evaluación Monte Carlo del sistema para obtener estimaciones de la tolerancia a los fallos.

El efecto del almacenamiento volátil puede acotarse considerando la probabilidad de que se produzca un fallo mientras haya mensajes inestables. Supongamos que el fallo temporal es un proceso de Poisson con tasa $J\backslash t$ y que los datos volátiles se vacían en el almacenamiento estable cada s unidades de tiempo. La probabilidad de que se produzca un fallo antes de la devolución de la escritura es

$$p = \frac{2e^{stt} + s2J\backslash 2 - 2sJ\backslash + 2}{t}$$

Para un valor típico de s = 30 segundos y 1/J\t = 15 días, p está tan cerca de cero que es despreciable.

4 Consistencia y retraso

Los protocolos de consistencia débil permiten que las réplicas contengan información desactualizada. Hay dos medidas relacionadas con este efecto: una relativa a la propagación de una actualización y otra a la consistencia de los valores replicados. El tiempo necesario para propagar una actualización de una réplica a otras muestra la rapidez con la que la información se pondrá a disposición de los clientes. La probabilidad de encontrar información desfasada, y la antigüedad de esta información, agrega los efectos de varias actualizaciones.

4.1 Modelado de simulación

Construimos dos modelos de simulación de eventos discretos del protocolo de antientropía con sello de tiempo: uno para medir el retraso de propagación y el otro para medir la edad de la información.

El simulador de retraso de propagación midió el tiempo necesario para que un mensaje de actualización, introducido en el momento cero, y sus acuses de recibo se propagaran a todas las réplicas disponibles. Utilizó dos eventos: la propagación anti-entropía y el fallo permanente del sitio. El simulador podía parametrizarse para utilizar diferentes políticas de selección de socios. El simulador se ejecutaba hasta que los intervalos de confianza del 95% eran inferiores al 5% o se habían procesado 10.000 actualizaciones. En la práctica, los intervalos de confianza del 95% se situaban generalmente entre el 1 y el 2%.

El simulador de la era de la información era más complejo. Utilizaba cinco eventos: uno para iniciar y otro para detener la simulación, uno para enviar mensajes de actualización, uno para realizar la antientropía y uno para leer datos de una réplica. La simulación se dejaba correr durante 1.000 unidades de tiempo para que alcanzara el estado estacionario. La simulación terminó a las 50.000 unidades de tiempo. En ningún caso la anchura del intervalo de confianza del 95% superó el 4% de la media. Los eventos de lectura, escritura y antientropía se modelaron como procesos de Poisson. El simulador incluía diferentes protocolos de selección de pareja y una multidifusión opcional no fiable en las escrituras.

El simulador mantenía dos estructuras de datos para cada réplica: el vector de resumen de antientropía y un número de versión de datos. También mantenía un contador global de versiones de datos. Los eventos de escritura incrementaban el contador de versión global y copiaban ese valor al número de versión de alguna réplica. Si se utilizaba una multidifusión no fiable, el número de versión se copiaba a otras réplicas si se recibía un datagrama no fiable simulado. Los eventos de antientropía propagaban los números de versión entre las réplicas, además de actualizar los vectores de resumen de las réplicas.

Los eventos de lectura se utilizaron para recoger medidas de la antigüedad esperada de los datos y la probabilidad de encontrar datos antiguos. Se seleccionó una réplica al azar y se comparó el número de versión de esa réplica con la versión global. La diferencia mostraba cuántas actualizaciones tenía que recibir la réplica.

4.2 Retraso de propagación

El tiempo necesario para propagar un mensaje de una réplica a otras fue la primera medida que investigamos. Si la información se propaga rápidamente, los clientes que utilicen diferentes réplicas no observarán a menudo información diferente, y la pérdida de una actualización por un fallo del sitio será poco probable. El tamaño del registro de mensajes está relacionado con esta medida, ya que los mensajes se eliminan del registro cuando se han recibido acuses de recibo de todas las réplicas.

La figura 3 muestra la probabilidad acumulada en el tiempo de que todas las réplicas hayan recibido una actualización. El tiempo se mide como múltiplos del intervalo medio en el que las réplicas inician los eventos de antientropía. Se supuso que la antientropía se produce 1.000 veces más a menudo que el fallo permanente del sitio -como hemos señalado, una estimación baja-. Las simulaciones de este gráfico

utilizan la selección aleatoria de parejas sin multidifusión inicial.

Esta figura también muestra que nuestros protocolos de consistencia débil se adaptan bien a un gran número de emplazamientos. El retraso de propagación aumenta aproximadamente como el logaritmo del número de sitios.

También consideramos el efecto de diferentes políticas de selección de socios en la velocidad de propagación. La figura 4 muestra la distribución para diferentes políticas, utilizando 500 sitios y sin multidifusión inicial. La selección aleatoria y la basada en la distancia son casi idénticas, mientras que la política de "primero el más antiguo" es, en general, algo

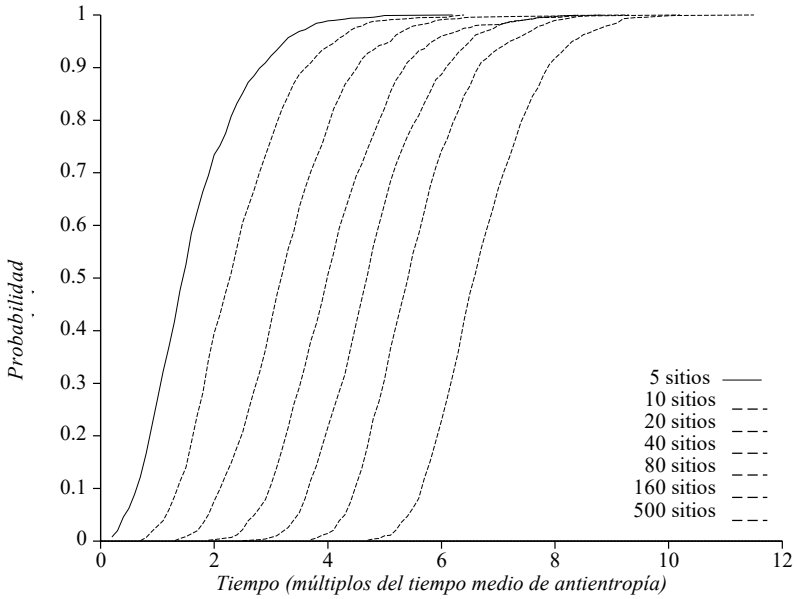

FIGURA 3: Distribución de la probabilidad acumulada de propagar un mensaje a todas las réplicas. Medido para la selección aleatoria de parejas con $p = 1000$

más lento que los demás. La velocidad de propagación de los acuses de recibo también es importante. Hemos comprobado que el retraso del acuse de recibo es igual de rápido que el de la propagación, y que la política de "primero el más antiguo" es más lenta que las otras dos.

4.3 La era de la información

Mientras que el retardo de propagación mide el rendimiento de una actualización, esta medida muestra la consistencia de las actualizaciones concurrentes en un solo elemento de datos. Tomamos dos medidas: la probabilidad de que una réplica devuelva información antigua, independientemente de su antigüedad, y la antigüedad esperada de la información, sea o no actual.

La edad de una entrada de la base de datos depende de la relación entre la tasa de antientropía y la tasa de actualización de esa entrada. Muchos servicios de área amplia tienen tasas de actualización extremadamente bajas; algunos servicios escriben nuevas entradas y nunca las cambian. Una tasa de actualización baja significa que la anti-entropía tiene más posibilidades de propagar una actualización antes de que otra entre en el sistema. En el Servicio de Nombres de Dominio [17], un nombre de host o una dirección rara vez cambia más de una vez cada pocos meses. En otras bases de datos se añaden nuevas entradas, se corrigen rápidamente y luego se mantienen estables. Esperamos que la tasa de actualización de una sola entrada de la base de datos sea unas mil veces menor que la tasa de antientropía. La mayoría de los gráficos de esta sección se generaron utilizando un tiempo medio de actualización de 1.000 unidades de tiempo; la tasa máxima de antientropía investigada fue sólo 200 veces mayor, lo que da un tiempo medio de antientropía de cinco.

La figura 5 muestra la probabilidad de obtener información desfasada, mientras que la figura 6 muestra la antigüedad esperada de esa información. Está claro que añadir una multidifusión no fiable en escritura mejora significativamente ambas medidas. Hemos comprobado que la probabilidad de éxito de los mensajes es la que más influye en la antigüedad de la información en grupos grandes de réplicas. Para números pequeños de sitios, aumentar la tasa de antientropía mejora drásticamente tanto la probabilidad de obtener información actualizada como la edad esperada.

Las figuras 7 y 8 muestran cómo la consistencia depende del número de sitios. Para estas simulaciones, la tasa de antientropía se fijó en 100 veces la de las escrituras. Esperamos que este valor sea el típico para una base de datos

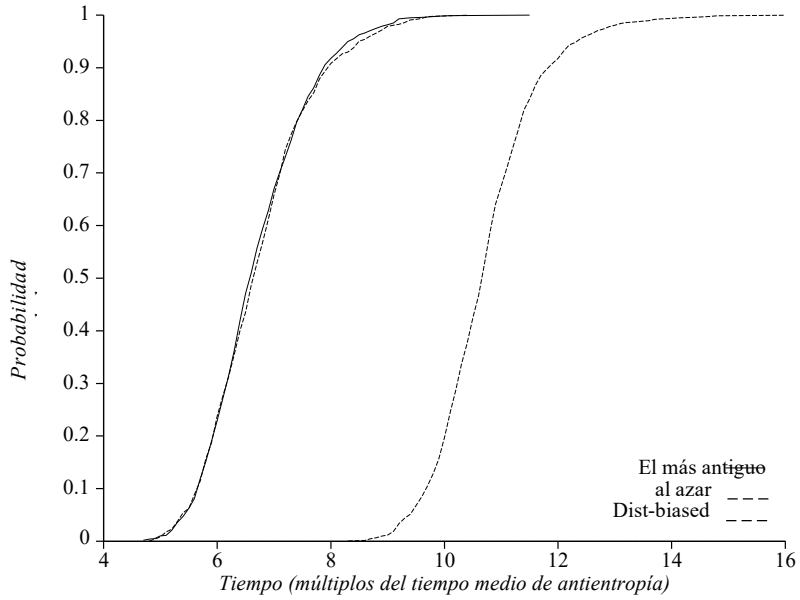

FIGURA 4: Efecto de la política de selección de socios en el retraso de propagación. Medido para 500 sitios con $p = 1000$

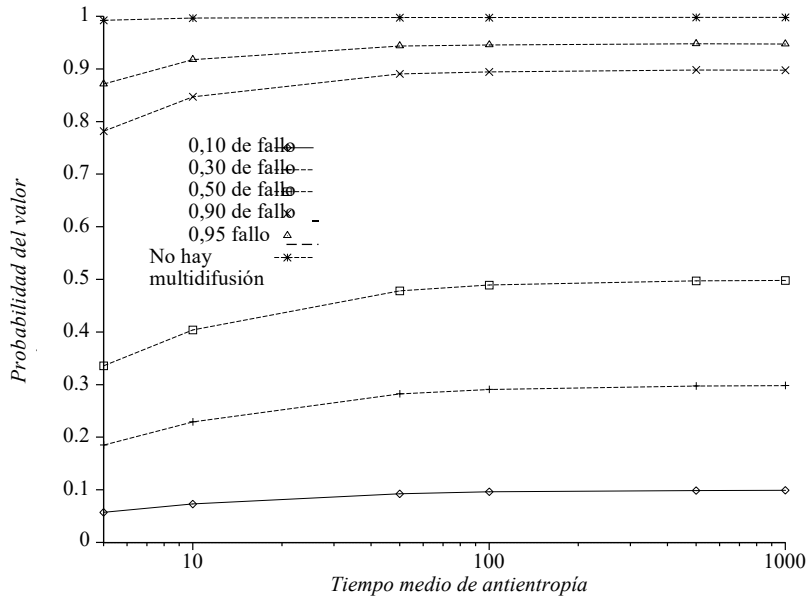

FIGURA 5: Probabilidad de obtener un valor antiguo al variar la tasa de antientropía, para 500 sitios. Tiempo medio de escritura 1 000; selección aleatoria de la pareja.

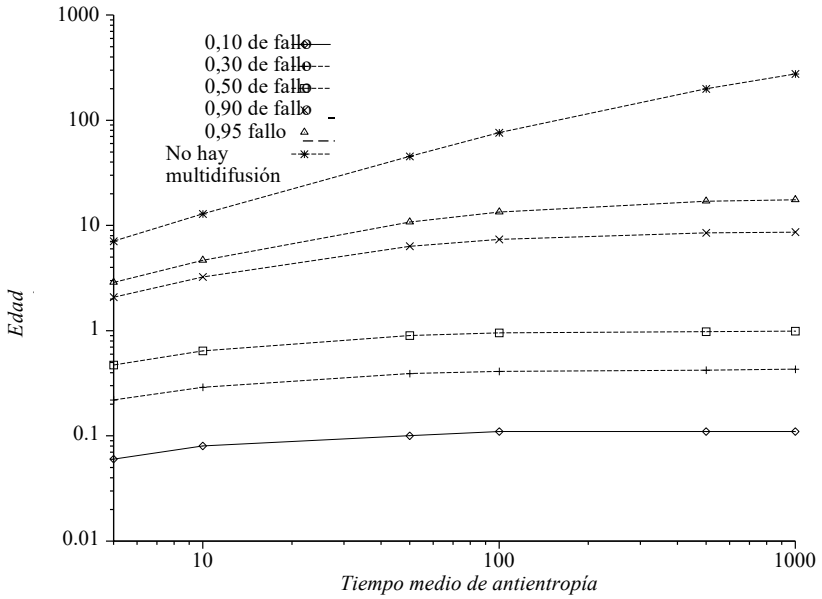

FIGURA 6: Edad esperada de los datos según varía la tasa de antientropía, para 500 sitios. Tiempo medio de escritura 1 000; selección aleatoria de la pareja.

poco después de que se introduzca, cuando las actualizaciones son más probables. Las actualizaciones posteriores serán menos frecuentes y la proporción aumentará, mejorando la consistencia. Una vez más, una multidifusión no fiable proporciona una mejora considerable.

También investigamos el efecto de la política de selección de pareja en la edad de la información, como se muestra en la figura 9. Dado que la selección aleatoria y la basada en la distancia mostraron tasas de propagación idénticas, no es de extrañar que tengan la misma edad esperada. La selección del más antiguo proporciona información ligeramente más antigua, lo que es coherente con una tasa de propagación más lenta.

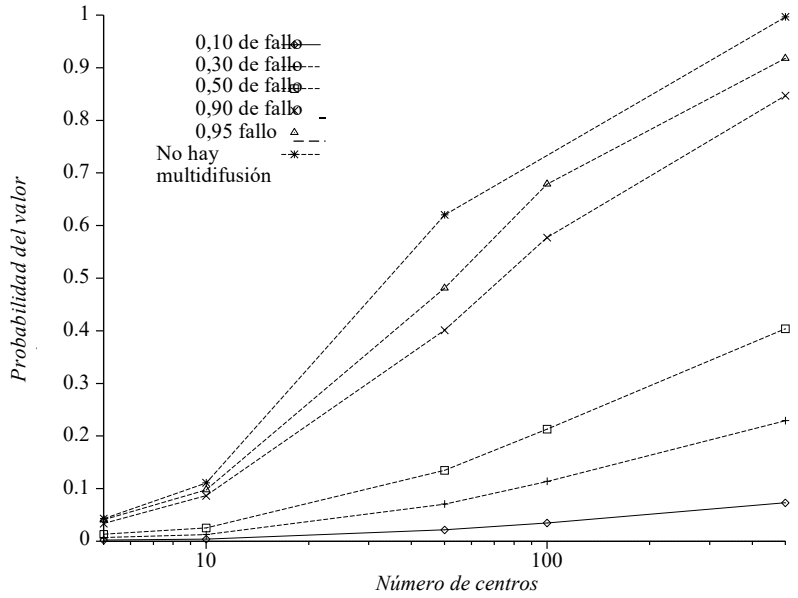

FIGURA 7: Probabilidad de obtener un valor antiguo a medida que varía el número de sitios, con la antientropía ocurriendo 100 veces más a menudo que las escrituras. Selección aleatoria de la pareja.

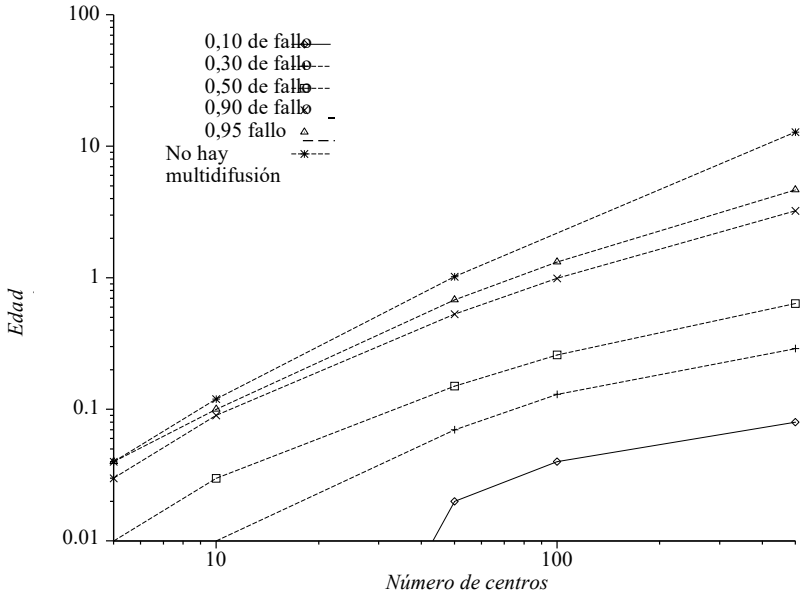

FIGURA 8: Edad esperada de los datos según varía el número de sitios, con una antientropía que ocurre 100 veces más a menudo que las escrituras.

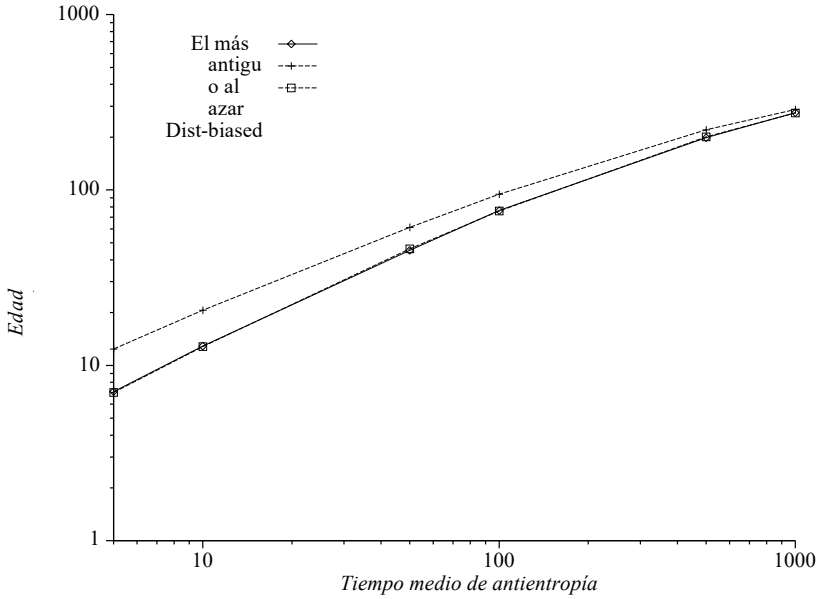

FIGURA 9: Efecto de la política de selección de socios en la antigüedad esperada de los datos. 500 sitios; tiempo medio para escribir 1 000.

5 Conclusiones

La escalabilidad, la tolerancia a los fallos y la compatibilidad con la informática móvil son algunos de nuestros objetivos para una arquitectura de servicios distribuidos de área amplia. Los protocolos de votación no pueden cumplir estos objetivos en una red de área amplia, por lo que hemos adoptado protocolos de replicación de consistencia débil.

Hemos comprobado que el protocolo antientropía con marca de tiempo se adapta bien a un gran número de réplicas. El tiempo necesario para propagar una actualización desde una réplica a todas las demás aumenta como el logaritmo del tamaño del grupo. Cada réplica sólo necesita almacenar *O(n)* estado en el tamaño del grupo.

La tolerancia a los fallos es la capacidad de un sistema de proporcionar un servicio correcto incluso cuando algunas partes fallan. El protocolo antientropía con marca de tiempo evita la comunicación sincrónica, comunicándose en cambio entre pares de réplicas. Cuando un sitio se separa del resto de la red, puede seguir prestando servicio y recibirá información actualizada una vez que se vuelva a conectar. Asimismo, no se requieren protocolos especiales para la recuperación tras un fallo temporal.

Estas mismas características hacen que la antientropía con sello de tiempo sea ideal para los sistemas informáticos móviles. Los usuarios pueden colocar réplicas en sus ordenadores portátiles. Los protocolos pueden hacer frente al gran número de réplicas que esto produce, y se asegurarán de que se actualicen cuando se conecten a la red.

Esto puede contrastarse con los protocolos de replicación consistentes, como la votación, que requieren una comunicación sincrónica con la mayoría de las réplicas. Mientras que los protocolos consistentes pueden proporcionar una buena disponibilidad y rendimiento con un pequeño número de réplicas, no son prácticos para cientos o miles de réplicas. Las réplicas consistentes no pueden seguir funcionando cuando se desconectan de otras réplicas, por lo que no son útiles para los sistemas informáticos móviles.

El aspecto negativo de los protocolos de consistencia débil es que las réplicas pueden devolver información desactualizada. Nuestra investigación descubre que una multidifusión poco fiable puede mitigar la mayor parte de este problema, y que a una velocidad de propagación razonable las réplicas rara vez se retrasan más que unas pocas actualizaciones. Muchas aplicaciones, como los servicios de nombres y las bases de datos bibliográficas, no se preocupan por la consistencia.

Nos sentimos alentados por el rendimiento de la política de selección de socios con sesgo de distancia. Políticas similares pueden utilizarse en Internet para fomentar el tráfico entre sitios cercanos y evitar la saturación de los enlaces de larga distancia. La política aleatoria parece estar a un factor constante del óptimo [18]. Estamos investigando su optimalidad, así como otras políticas de selección.

Agradecimientos

John Wilkes, del Proyecto de Sistemas Concurrentes de los Laboratorios Hewlett-Packard, y Kim Taylor, de la UC Santa Cruz, colaboraron en el desarrollo inicial de estos protocolos. George Neville-Neil, Bruce Montague y Mary Long aportaron útiles comentarios a este documento.

Las soluciones analíticas se obtuvieron con la ayuda de *Maple,* un programa de álgebra simbólica desarrollado por el Symbolic Computation Group de la Universidad de Waterloo. Muchos de los resultados de la simulación se obtuvieron con la ayuda de SIMSCRIPT II.5, un lenguaje de simulación desarrollado y apoyado por CACI Products Company de La Jolla, California.

Richard Golding ha sido financiado en parte por el Proyecto de Sistemas Concurrentes de los Laboratorios Hewlett-Packard y por una beca de postgrado de la Operación Santa Cruz. Darrell Long ha recibido parte del apoyo de la National Science Foundation con la subvención NSF CCR-9111220.

Referencias

[1] J. S. Quarterman y J. C. Hoskins, "Notable computer networks", *Communications of the ACM*, vol. 29, pp. 932-71, octubre de 1986.

[2] M. D. Schroeder, A. D. Birrell y R. M. Needham, "Experience with Grapevine: the growth of a distributed system", *ACM Transactions on Computer Systems*, vol. 2, pp. 3-23, febrero de 1984.

[3] R. A. Golding, "Accessing replicated data in a large-scale distributed system", tesis de máster, Computer and Information Sciences Board, University of California at Santa Cruz, junio de 1991. Publicada como Tech. Rep. UCSC-CRL-91-18.

[4] D. D. E. Long, J. L. Carroll y C. J. Park, "A study of the reliability of Internet sites", en *Proceedings of 10th IEEE Symposium on Reliable Distributed Systems*, pp. 177-86, septiembre de 1991.

[5] A. Emtage y P. Deutsch, "archie - an electronic directory service for the Internet", en *Proceedings of Winter 1992 Usenix Conference*, pp. 93-110, enero de 1992.

[6] J. J. Kistler y M. Satyanarayanan, "Disconnected operation in the Coda file system", en *Proceedings of 13th ACM Symposium on Operating Systems Principles*, pp. 213-25, octubre de 1991.

[7] R. Alonso, D. Ba y L. L. Cova, "Using stashing to increase node autonomy in distributed file systems", en *Proceedings of 9th IEEE Symposium on Reliable Distributed Systems*, octubre de 1990.

[8] L. Lamport, "Time, clocks, and the ordering of events in a distributed system", *Communications of the ACM*, vol. 21, no. 7, pp. 558-65, 1978.

[9] R. Ladin, B. Liskov, y L. Shrira, "Lazy replication: exploiting the semantics of distributed services," *Operating Systems Review*, vol. 25, pp. 49-55, enero de 1991.

[10] C. Pu y A. Leff, "Replica control in distributed systems: an asynchronous approach", Tech. Rep. CUCS-053-090, Department of Computer Science, Columbia University, enero de 1991.

[11] D. Barbara' y H. García-Molina, "The case for controlled inconsistency in replicated data", en *Actas del taller sobre la gestión de datos replicados*, pp. 35-8, noviembre de 1990.

[12] J. Turek y D. Shasha, "The many faces of consensus in distributed systems", *IEEE Computer*, vol. 25, pp. 8-17, junio de 1992.

[13] R. A. Golding, "The timestamped anti-entropy weak-consistency group communication protocol", Tech. Rep. UCSC-CRL-92-29, Computer and Information Sciences Board, University of California at Santa Cruz, julio de 1992.

[14] R. A. Golding y K. Taylor, "Group membership in the epidemic style", Tech. Rep. UCSC-CRL- 92-13, Computer and Information Sciences Board, University of California at Santa Cruz, abril de 1992.

[15] D. Agrawal y A. Malpani, "Efficient dissemination of information in computer networks", *Computer Journal*, vol. 34, pp. 534-41, diciembre de 1991.

[16] A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart y D. Terry, "Epidemic algorithms for replicated database maintenance", *Operating Systems Review*, vol. 22, pp. 8-32, enero de 1988.

[17] P. Mockapetris, "Domain names - concepts and facilities", Tech. Rep. RFC 1034, ARPA Network Working Group, noviembre de 1987.

[18] N. Alon, A. Barak y U. Manber, "On disseminating information reliably without broadcasting", en *Actas de la 7ª Conferencia Internacional sobre Sistemas Informáticos Distribuidos*, pp. 74-81, 1987.