Casa abierta al tiempo
Universidad Autónoma Metropolitana

Evaluación de Estrategias de Redundancia de Información en Sistemas de Almacenamiento P2P

Idónea comunicación de resultados para obtener el grado de

MAESTRO EN CIENCIAS
(CIENCIAS Y TECNOLOGÍAS DE LA INFORMACIÓN)

PRESENTA
Alan Gustavo Lazalde Cruz

ASESOR
Dr. Ricardo Marcelín Jiménez

25 de noviembre de 2009

Casa abierta al tiempo
Universidad Autónoma Metropolitana

Evaluación de Estrategias de Redundancia de Información en Sistemas de Almacenamiento P2P

Idónea comunicación de resultados para obtener el grado de

Maestro en Ciencias
(Ciencias y Tecnologías de la Información)

Presenta
Alan Gustavo Lazalde Cruz

Asesor
Dr. Ricardo Marcelín Jiménez

25 de noviembre de 2009

Resumen

En la práctica y en la literatura pueden hallarse diversas metodologías para redundar información en un sistema de almacenamiento distribuido, que van desde la simple copia fiel de los objetos de información hasta el uso de sofisticadas *técnicas de codificación* (e.g., códigos tipo MDS, códigos sin tasa, códigos de red).

Para efectos de nuestro trabajo de investigación, *evaluamos* un conjunto representativo de esas maneras de redundar información bajo una serie de parámetros básicos, con el fin de ofrecer al interesado —e.g., el diseñador de un sistema de almacenamiento distribuido— una *guía* detallada con los elementos que le ayuden a decidir cuál de ellas es la más apropiada para el contexto de su implementación, con especial énfasis en los sistemas de almacenamiento P2P (*Peer-to-Peer*).

Veremos que, en última instancia, la elección de una estrategia de redundancia frente a otra dependerá de los compromisos entre diversos factores, los cuales no sólo tienen origen en las estrategias mismas sino también en las restricciones impuestas por los nodos que participan en la red de almacenamiento. Así pues, nuestro trabajo es una *visión general* de esos compromisos.

Contenido

Lista de figuras

Lista de tablas

Lista de símbolos

Símbolo	Significado
A	Disponibilidad de la información.
$\mathbf{A}$	Matriz de coeficientes.
$\hat{\mathbf{A}}$	Matriz inversa de coeficientes, tal que todos sus renglones son linealmente independientes.
$\mathbf{a}_i$	Vector i-ésimo de la matriz de coeficientes $\mathbf{A}$.
a	Disponibilidad promedio de los nodos de una red.
$a_{i,j}$	Coeficiente de la matriz $\mathbf{A}$.
$\alpha_{i,j}$	Coeficiente de la matriz $\hat{\mathbf{A}}$.
b	Cantidad de bits por símbolo $s_{i,j}$.
$\overline{b}$	Ancho de banda promedio por nodo debido a mantenimiento de la redundancia.
$\mathbf{c}$	Palabra código.
$C(X^j)$	Conjunto de nodos a los que el nodo X^j está conectado en $\mathscr{F}$.
$\mathbf{d}_i$	Vector que representa un disperso.
$\overline{d}$	Almacenamiento promedio por nodo.
$\mathcal{D}$	Cantidad de bytes únicos en el sistema de almacenamiento.
$\mathcal{D}_t$	Almacenamiento total en el sistema luego de aplicar redundancia a $\mathcal{D}$.
$\mathbf{D}$	Matriz de dispersos.
$\mathbf{e}$	Error generado por un canal de comunicaciones.
ϵ	Exceso de dispersos para recuperar $\mathbf{F}$.
E	Conjunto de aristas de un grafo $\mathscr{G}$.
f_i	Pieza i-ésima del archivo F.
$\mathbf{f}_i$	Vector i-ésimo de la matriz $\mathbf{F}$, y representa a f_i.
F	Archivo u objeto de información compuesto de símbolos que pertenecen a $\mathbb{F}_p$.
$\mathbf{F}$	Matriz que representa al archivo F.
$\mathbb{F}_p$	Campo finito de orden p.
$\mathcal{F}$	Número de objetos de información en un sistema de almacenamiento.

Símbolo	Significado
$\mathscr{F}$	Grafo de flujo de información que representa un sistema de almacenamiento distribuido.
$\mathscr{G}$	Red de flujo.
h	Nodos a los que se conecta un nodo recién llegado en $\mathscr{F}$.
i, j	Índices.
k	Factor de estiramiento o redundancia, para estrategias con códigos o sin ellos, respectivamente.
λ	Tasa de arribo de nodos al sistema de almacenamiento.
m	En general, es la cantidad símbolos o unidades necesarias para recuperar la información original. También es el tamaño de las unidades $\mathbf{f}_i$.
$\mathbf{m}$	Mensaje original transmitido por un canal con errores.
M	Tamaño promedio de los objetos de información.
μ	Tasa de partida de nodos del sistema de almacenamiento.
n	En general, es la cantidad de objetos de información distribuidos a los largo del sistema de almacenamiento. También es la cantidad de símbolos que resultan de una codificación.
N	Cantidad de símbolos en F.
$\mathcal{N}$	Número de nodos en el sistema de almacenamiento.
ω	Fracción de información que descarga un nodo recién llegado en $\mathscr{F}$.
Ω	Cantidad de información que almacena un nodo en $\mathscr{F}$.
p	Número primo.
π	Ruta de aumentación.
ρ	Probabilidad de transmitir un mensaje sin errores.
$\mathbb{R}$	Conjunto de los números reales.
s	Vértice fuente en $\mathscr{G}$.
$s_{i,j}$	Símbolo que pertenece a un campo finito $\mathbb{F}_p$.
$\mathcal{S}$	Desviación estándar muestral.
t	Vértice sumidero en $\mathscr{G}$.
t_i	Estampilla de tiempo asignada a un nodo a su llegada o salida del sistema de almacenamiento.
T	Periodo de permanencia de un nodo en el sistema de almacenamiento.
τ	Tiempo de expiración de membresía de un nodo en el sistema de almacenamiento.
u, v	Vértices en $\mathscr{G}$.
V	Conjunto de vértices del grafo $\mathscr{G}$.
$\overline{X}$	Desviación estándar muestral.
$X^i_{in} \xrightarrow{\Omega} X^i_{out}$	Nodo i-ésimo en $\mathscr{F}$ con capacidad de almacenamiento Ω bytes.
$\mathbb{Z}$	Conjunto de los número enteros.
$\mathbb{Z}_p$	Conjunto de enteros $\mod -\, p$.

Lista de acrónimos

Acrónimo	Significado
DAS	Almacenamiento por conexión directa.
DC	Colector de datos en $\mathscr{F}$.
DHT	Tabla *hash* distribuida.
DS	Fuente de datos en $\mathscr{F}$.
IDA	Algoritmo de dispersión de información.
iSCSI	Almacenamiento SCSI por internet.
NAS	Almacenamiento por conexión en red.
NFS	Sistema de archivos en red.
MDS	Códigos separados por distancia máxima.
MDT	Tiempo medio de no funcionamiento.
MTBF	Tiempo medio entre fallas.
MTTF	Tiempo medio hasta fallar.
MTTR	Tiempo medio hasta recuperar.
MUT	Tiempo medio de funcionamiento.
P2P	*Peer-to-Peer*.
RB	Redundancia por bloques.
RCR	Redundancia por códigos de red.
RFD	Redundancia por fuente digital.
RH	Redundancia híbrida.
RIDA	Redundancia por IDA.
RS	Redundancia simple.
SAN	Área de almacenamiento por red.
SMTP	Protocolo simple de transferencia de correo.
TCP/IP	Protocolos de control de transporte e internet.
URI	Identificador uniforme de recursos.

Capítulo 1

Introducción

Desde sus inicios, los sistemas *Peer-to-Peer* (P2P [1]) se convirtieron en punto de referencia para una gran cantidad de trabajos de investigación, los cuales buscaron —y lograron— mejorar su desempeño, básicamente en términos de escala, autonomía y autoorganización, así como aumentar la gama de aplicaciones P2P.

Hoy en día, los sistemas P2P se implementan sobre un rango muy amplio tecnologías de cómputo distribuido. Algunos ejemplos de esto son los juegos en línea, el cómputo científico, la colaboración distribuida, la mensajería instantánea y la distribución de archivos.

Por otro lado, el hecho de que los dispositivos de almacenamiento mejoraran su capacidad y desempeño durante los últimos años, sumado a una dinámica de rápido crecimiento en la cantidad y calidad de aplicaciones P2P, abrió las puertas a una nueva clase de sistemas de cómputo: los llamados *sistemas de almacenamiento P2P*.

La calidad de un sistema de almacenamiento de cualquier tipo está determinada por diversos factores: ancho de banda, espacio de almacenamiento, latencia, tiempo de respuesta, escalabilidad, disponibilidad. De todos ellos, para este trabajo consideramos que la *disponibilidad de la información* es el más relevante porque tiene que ver directamente con el propósito fundamental de todo sistema de almacenamiento: *resguardar y entregar información*. ¿Cómo asegurar que la información estará disponible en un sistema de almacenamiento compuesto por una red P2P, donde sus componentes son libres de todo control central? Como en cualquier sistema de almacenamiento tradicional, la clave para conseguir los grados de disponibilidad deseados está en la *redundancia de información*.

En la práctica y en la literatura se pueden encontrar varias maneras de redundar información, que van desde la simple copia fiel de los objetos de información hasta el uso de sofisticadas técnicas de codificación. El *objetivo* de este trabajo de investigación es *evaluar* un conjunto representativo de esas maneras de redundar información —*estrategias*— bajo una serie de parámetros básicos, con el fin de ofrecer al lector —e.g., el diseñador de un sistema de almacenamiento distribuido, en general; P2P, en particular— una *guía* detallada con elementos que le ayuden a decidir cuál de ellas es más apropiada para el contexto de su implementación.

Veremos que, en última instancia, la elección de una estrategia de redundancia frente a otra dependerá de los compromisos entre diversos factores, los cuales no sólo tienen origen en las estrategias mismas sino también en las restricciones impuestas por los nodos que participan en la red P2P. Este trabajo es una *visión general* —un *survey*— de esos compromisos.

Este documento se encuentra dividido en tres partes. En la primera parte (§ I) ofrecemos los antecedentes teóricos que están relacionados con la problemática del almacenamiento en redes P2P. Esta visión va de lo general a lo particular, comenzando por definir los parámetros que nos hablan del desempeño de

[1] Existen diversas traducciones al castellano del término *peer-to-peer*: *entre pares*, *igual a igual*, *entre iguales*, *entre homólogos*, etc; prefiriéndose la primera. No obstante, para este documento mantendremos la sigla P2P y su pronunciación en inglés, puesto que es de uso muy extendido y no da lugar a confusiones.

un sistema de cómputo (§ 2.1) y los sistemas de almacenamiento (§ 2.2). Sobre las maneras de redundar la información para ser transportada, almacenada y recuperada, con aspectos básicos de teoría de la información, códigos y las redes de flujo (§ 2.3). Asimismo, describimos las características que definen los sistemas P2P (§ 3.1), con énfasis en su rol como sistemas de almacenamiento. Con todo lo anterior se fundamenta y muestra una visión general basada en etapas (§ 3.3) de la forma en que cada estrategia de redundancia puede operar en sistemas P2P (§ 3.4).

La segunda parte de este trabajo (§ II) comienza con una breve revisión del trabajo previo relacionado con la evaluación de estrategias de redundancia en sistemas de almacenamiento P2P (§ 4). Nuestra metodología de evaluación (§ 5). Acto seguido, ofrecemos análisis y resultados de nuestra evaluación de seis estrategias de redundancia de información bajo tres parámetros: factor de redundancia (§ 6), almacenamiento promedio por nodo (§ 7) y ancho de banda por mantenimiento (§ 8).

En la tercera y última parte (§ III) cerramos con nuestras conclusiones generales (§ 9) y una lista de recomendaciones de trabajo futuro (§ 9), que pueden servir como punto de partida en otros proyectos de investigación.

Parte I

Antecedentes

Capítulo 2

Almacenamiento de la Información

Los sistemas de almacenamiento de información son sistemas de cómputo que como tales son suceptibles de fallar o degradarse. ¿Cómo medir y evaluar objetivamente la *calidad* de dichos sistemas? Esto se logra mediante parámetros que ayuden a cuantificar y calificar cada sistema en relación con el servicio que de ellos esperamos, es decir, mediante la *evaluación de su desempeño*. Las respuestas a esa pregunta fundamentan los resultados obtenidos en este trabajo de investigación y la conformación final de una *guía* para los diseñadores de sistemas de almacenamiento.

En las siguientes secciones exploramos las características de los sistemas de cómputo para luego continuar con aquellos especializados en el almacenamiento de la información. Estos, vistos a la luz de parámetros para medir y evaluar cómo cumplen con su tarea, con especial énfasis en el que llamaremos *disponibilidad*.

2.1. Sistemas de cómputo

Un sistema de cómputo es una colección de elementos de hardware, software de base y software aplicativo, quizá colocados en un ambiente de red [18], que realiza tareas de procesamiento de datos. Donde tal ambiente consiste de aquellos parámetros externos al sistema y de sus variaciones en el tiempo. En cuanto a las responsabilidades del sistema, éste se encuentra limitado por la manera como ofrece sus servicios, donde un servicio puede verse desde distintos niveles de abstracción. Existen dos tipos de servicio, que son:

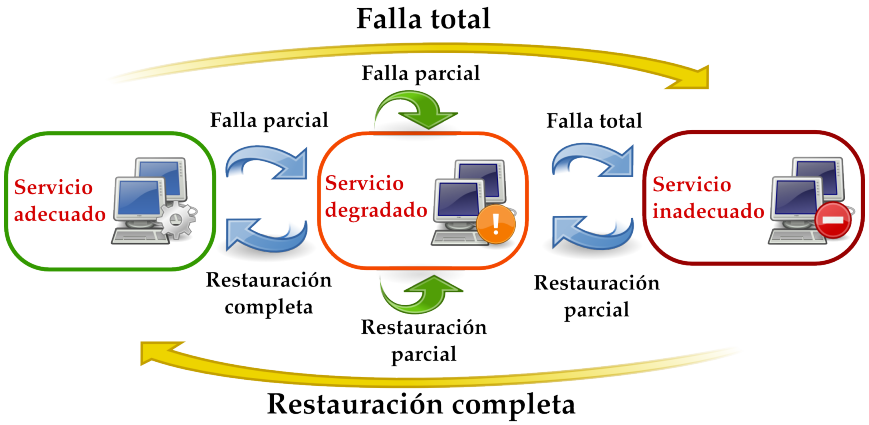
- *Servicio esperado.* Los resultados y las condiciones bajo las cuales son entregados al usuario en un ambiente dado, según está dictado en la especificación del sistema.
- *Servicio provisto.* Los resultados y las condiciones efectivas bajo las cuales son entregados.

Desde el punto de vista del usuario y los servicios que se le ofrecen, un sistema de cómputo puede encontrarse en uno de los siguientes tres estados:

- *Servicio adecuado.* Cuando el sistema ofrece el servicio esperado.
- *Servicio degradado*. El sistema ofrece el servicio esperado, excepto en la manera de entregar los resultados.
- *Servicio inadecuado*. El sistema no ofrece el servicio esperado.

La Fig. 2.1 es un diagrama de los estados de un sistema. Nótese que las transiciones que se alejan del estado *servicio adecuado* incluyen la palabra *falla*. En el contexto de los sistemas, una falla es una discordancia entre el servicio provisto y el servicio esperado, y que puede ser detectada por el usuario o por el sistema mismo.

Figura 2.1: *Estados del sistema: servicios y fallas* [18].



Las fallas tienen grados. Una falla completa ocurre cuando el sistema ofrece servicios inadecuados; e.g., cuando un sistema operativo se bloquea. Una falla es parcial cuando el sistema ofrece un servicio degradado; e.g., cuando un sistema operativo es demasiado lento para ofrecer sus servicios. El concepto de falla es indispensable para entender la *disponibilidad* de un sistema de cómputo, un concepto que abordaremos más adelante.

2.1.1. Medidas de desempeño

Para cuantificar el desempeño de un sistema de cómputo consideraremos algunas medidas[1] que a su vez están relacionadas con diferentes conceptos: fallas, mantenibilidad, disponibilidad y servicio del sistema [18, 84]. Las medidas relacionadas con las fallas y la mantenibilidad nos hablan de la confiabilidad que ofrece el sistema.

Medidas relacionadas con las fallas

- $MTTF$ (*Tiempo medio hasta fallar*). Es el valor esperado[2] de la vida operacional del sistema hasta la primera falla, i.e., es la cantidad de tiempo que se espera que el sistema trabaje sin una falla. Este valor se estima como la suma acumulativa de intervalos de tiempo libres de falla dividido por el número de fallas observadas en el intervalo de tiempo de observación.

- $MTBF$ (*Tiempo medio entre fallas*). Es el valor esperado de la duración entre dos fallas. Igual que $MTTF$, se estima sumando acumulativamente los intervalos de tiempo libres de falla, más la suma acumulativa del tiempo necesario para restaurar el sistema hasta que ofrezca un servicio adecuado, dividido todo por el número de fallas que hubo durante el tiempo de observación.

Medidas relacionadas con las mantenibilidad

- $MTTR$ (*Tiempo medio hasta recuperar*). Es el valor esperado del tiempo hasta que el sistema sea reparado y pueda comenzar a utilizarse de nuevo. La reparación del sistema puede llevarse cabo reemplazado el componente defectuoso. Se estima como la suma acumulativa de los tiempos necesarios

[1] Tales medidas deben tener las siguientes propiedades: representativas del fenómeno de estudio, interpretables, consistentes, precisas y útiles [18].

[2] Aquí decimos *esperado* porque se trata de la *esperanza matemática* de una variable aleatoria, y no del promedio aritmético de un conjunto de sucesos.

Tabla 2.1: *Clasificación de sistemas en función de su disponibilidad* [35].

Tipo de sistema	No disponible (*min./año*)	Disponibilidad	Clase
No administrado	50,000	90.0 %	1
Administrado	5,000	99.0 %	2
Bien administrado	500	99.9 %	3
Tolerante a fallas	50	99.99 %	4
Alta disponibilidad	5	99.999 %	5
Muy alta disponibilidad	0.5	99.9999 %	6
Ultra alta disponibilidad	0.05	99.99999 %	7

para la reparación del sistema dividida por el número de veces que ocurrió una reparación durante el tiempo de observación [84, 83].

Medidas relacionadas con el servicio

- MUT (*Tiempo de funcionamiento medio*). Esperaza matemática de la duración en tiempo en el que el sistema provee el servicio esperado durante un tiempo de observación dado. Esto es equivalente a dividir la cantidad del tiempo que el sistema trabajó en un estado adecuado entre el número de veces que pasó a un estado no adecuado, durante el intervalo de tiempo de la observación.

- MDT (*Tiempo de no funcionamiento medio*). De manera análoga a MUT, se trata de la esperanza matemática de la duración en tiempo en el que el sistema *no* provee el servicio esperado durante un tiempo de observación dado. Y que es equivalente a dividir la cantidad del tiempo que el sistema *no* trabajó en un estado adecuado entre el número de veces que pasó a ese estado, durante el intervalo de tiempo de la observación.

2.1.2. Disponibilidad

Cada día un número creciente de actividades de diversos sectores como gobierno, salud, investigación y negocios, basa sus actividades en el procesamiento de información en sistemas de cómputo. Para ellos la disponibilidad de sus recursos puede llegar a ser de importancia crítica. Particularmente, el control de procesos, control de producción y las aplicaciones de procesamiento de transacciones, demandan los niveles de disponibilidad más altos, i.e., *alta disponibilidad* [35]. Estos se encuentran en las redes telefónicas, los sistemas de tráfico aéreo, hospitales, fábricas, y más. Si los recursos de cómputo requeridos no están disponibles puede significar retrasos en la producción y la consecuente pérdida de dinero, equipo dañado, y en el peor de los casos, pérdida de vidas.

Asegurar la disponibilidad de los sistemas de cómputo, es también asegurar la de los recursos que ofrecen. Esta es una tarea compleja que ha merecido una gran cantidad de trabajos de investigación. Algunos de ellos datan desde el principio mismo de los sistemas de cómputo, cuando el hardware y no el software, como sucede en la actualidad, era más propenso a fallar.

La disponibilidad de un sistema puede clasificarse según los minutos que se encuentra en funcionamiento durante un año (Tabla 2.1). Informalmente, el número de cifras «9» en el porcentaje de disponibilidad indica la clase del sistema. Formalmente, dada una disponibilidad A del sistema, entonces [35]:

$$clase = e^{log_{10}(\frac{1}{1-A})}.$$

Así pues, definimos un sistema con alta disponibilidad como el que permite, a los más, cinco minutos de interrupción de servicio por año, i.e., 99.999 % de disponibilidad. En general, la disponibilidad de un sistema es una característica básica de los llamados *sistemas confiables*; las otras características son [18]:

- *Confiabilidad.* Una medida de la continuidad de los servicios entregados.
- *Mantenibilidad.* Habilidad de un sistema para mantenerse en condiciones de operación, y para ser reparado y modificado.
- *Seguridad.* Estado donde existe carencia de fenómenos que provoquen consecuencias indeseables en el sistema y su entorno.
- *Inmunidad.* Capacidad del sistema para resistir ataques externos, accidentales o deliberados.

De manera más formal $Disponibilidad$ es la probabilidad, expresada como un porcentaje, de que el servicio esté disponible en cualquier momento dentro de límites aceptables de desempeño y sin fallas irrecuperables; i.e., es el grado en que el sistema opera de manera adecuada. Conociendo $MTTF$, $MTTR$ o $MTBF$, la $disponibilidad$ está dada por alguna de las siguientes fórmulas:

$$disponibilidad = \frac{MTTF}{MTBF} = \frac{MTTF}{MTTF + MTTR} = \frac{MTBF}{MTBF + MTTR}$$

Cuando $disponibilidad = 1$ es el caso ideal, es decir, cuando $MTTR = 0$ porque no existieron fallas que pasaran al sistema a un estado no adecuado.

2.2. Sistemas de almacenamiento

Se sabe que la producción mundial de información aumenta rápidamente año con año, y es transportada en todo tipo de medios —e.g., papel, discos magnéticos y ópticos, películas, cintas— para todo tipo de contenidos —e.g., libros, periódicos, fotos, audio, vídeo—. Particularmente, la creación y almacenamiento de información en sistemas de cómputo se ha convertido en una actividad ubicua de la vida diaria. Este hecho también ha dado pie a cambios en la dinámica de cómo el almacenamiento en sistemas de cómputo es utilizado y accedido, implicando un crecimiento en los requerimientos de los usuarios. En general, podemos clasificar en tres categorías [18] las aplicaciones con diferentes requerimientos de almacenamiento, en función del uso intensivo de:

- *Datos*. La aplicación debe poder ofrecer acceso directo a todos los datos —generalmente, una gran cantidad— de manera independiente del medio de almacenamiento.
- *Operaciones de Lectura/Escritura*. La aplicaciones que realizan transacciones utilizan gran cantidad de operaciones de lectura/escritura y están limitadas por el desempeño del disco.
- *Cómputo*. Los requerimientos de almacenamiento son pocos porque la aplicación se concentra en explotar las capacidades del procesador.

El aumento en la capacidad de los discos duros, el incremento de las velocidades de lectura y escritura, la reducción de los tiempos de acceso al disco y las escalas cada vez menores de los dispositivos (Tabla 2.2), son factores que gracias a los avances en las tecnologías de almacenamiento, ayudaron a promover ese rápido crecimiento en la creación de datos e información.

Tabla 2.2: *Variables usadas por el modelo probabilístico.*

Característica	Factor
Capacidad	Aumento de 100×
Velocidad de Lectura/Escritura	Aumento de 10×
Tiempo de acceso	Reducción de 3×
Escala del dispositivo	Reducción de 5×

Por otro lado, también Internet es un componente cotidiano en el trabajo y en aspectos como la comunicación y el entretenimiento cotidiano. En el 2003 la Universidad de Berkeley estimó que, de acuerdo al volumen de información en la web, el tamaño de Internet en dicho año fue de 167 TB. La empresa Horison ofrece una compilación anual [58] de datos, hechos y estimaciones relacionadas con el almacenamiento en sistemas de cómputo.

En el año 2001, por ejemplo, el tamaño promedio de un correo-e, incluyendo archivos adjuntos, fue de 50 Kb; en 2007 este número creció a 650 KB. En este sentido vale la pena mencionar a Gmail, el servicio gratuito de correo electrónico de Google, porque fue el primero de su tipo en ofrecer 1 GB de espacio de almacenamiento, y que en relativamente poco tiempo extendió sus límites por varios gigabytes más. En comparación, en el 2004 Yahoo! ofreció a sus usuarios de 20 MB a 100 MB de espacio de almacenamiento, más que nadie hasta ese momento y mucho menos que Gmail pocos años después. No nos queda duda de que rápidamente Internet y las tecnologías de almacenamiento se intersectaron para revolucionar la forma de responder a las preguntas sobre el qué, cómo y dónde depositar datos e información.

2.2.1. Clasificación

Almacenamiento Centralizado

Llamaremos almacenamiento centralizado al que *concentra* la administración de los datos y la información en una sola localidad, que puede ser un servidor o una granja (*cluster*) de servidores, sin importar la tecnología de comunicaciones que lo sustenta ni la manera —i.e., el protocolo— en que entrega su servicio. Las arquitecturas de almacenamiento centralizado más utilizadas son [18]:

- **DAS (*Directly Attached Storage*)**

  Los recursos de almacenamiento están conectados directamente a la computadora y reservados, normalmente, para su uso exclusivo. Los datos son entregados a través de una interfaz física de entrada/salida por bloques; e.g., SCSI. La computadora accede a los datos mediante el sistema de archivos que tiene implementado (Fig. 2.2, izquierda).

- **SAN (*Storage Area Network*)**

  Es una arquitectura que adjunta dispositivos de almacenamiento remotos de manera tal que aparecen ante el cliente como si fuesen locales, y donde las operaciones de lectura y escritura del cliente se hace a nivel de bloques. Para lograr un desempeño satisfactorio los sistemas SAN se valen de redes locales de alta velocidad, donde prevalece el uso de tecnologías de red de fibra óptica.

  Básicamente, la diferencia entre las arquitecturas NAS y SAN es que la primera ofrece la funcionalidad de un sistema de archivos y la segunda no [31] (Fig. 2.2, izquierda).

Figura 2.2: *Arquitecturas DAS, SAN, iSCSI y NAS* [18].

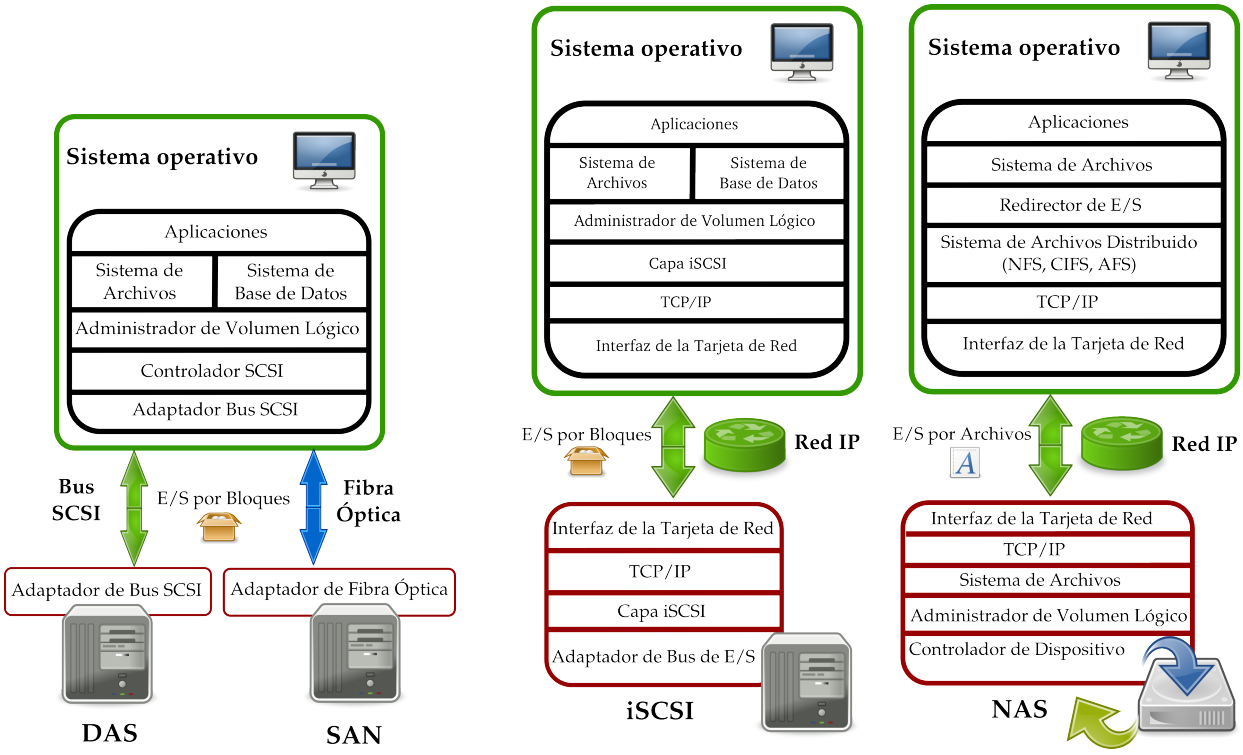


Tabla 2.3: *Comparación entre arquitecturas de almacenamiento centralizado* [18].

	Conexión Directa	Conexión en Red
Entrada/Salida por Bloques	DAS, SAN	iSCSI
Sistema de Archivos		NAS

- **iSCSI (*Internet SCSI*)**

  Esta arquitectura permite que clientes accedan a un sistema de almacenamiento remoto sobre Internet, a nivel de entrada/salida por bloques. iSCSI simplemente implementa los protocolos SCSI sobre la pila de protocolos TCP/IP, así que puede verse como una arquitectura SAN sobre TCP/IP (Fig. 2.2, centro).

- **NAS (*Network Attached Storage*)**

  Quizá una de las arquitecturas más utilizadas por la facilidad de su implementación. En ella los recursos de almacenamiento se distribuyen a los clientes a través de la red y accedidos por ellos mediante sistemas de archivos por red; e.g., NFS, CIFS. Esto permite a los clientes leer y escribir datos como si fueran operaciones locales, aunque son remotas (Fig. 2.2, derecha).

La Tabla 2.3 resume las características de las cuatro arquitecturas de almacenamiento centralizado presentadas arriba.

Almacenamiento Distribuido

Una de las grandes desventajas de los sistemas de almacenamiento centralizado es el crecimiento de los costos de mantenimiento en proporción al crecimiento en la escala del sistema. Por ejemplo, el almacenamiento y procesamiento de la cantidad de información generada por algunos centros de investigación —actualmente, del orden de petabytes— es prácticamente incosteable de manera centralizada, más aún cuando esos datos deben compartirse entre ellos. La opción natural ante esa problemática es distribuir las responsabilidades de almacenamiento.

El almacenamiento distribuido es una técnica utilizada para almacenar información a lo largo de múltiples nodos en la red. La distribución de los datos sobre una red añade requerimientos al sistema de almacenamiento, principalmente en términos de la escala, privacidad, anonimato, balance de carga, entre otros [92]. En [67] ofrecen una revisión de los sistemas de almacenamiento distribuido desde nueve diferentes temáticas, a cada una de las cuales le conceden una taxonomía propia. En este trabajo sólo mencionaremos dos de esas nueve clasificaciones:

1. *Propósito del sistema*. Sobre la función primordial del sistema de almacenamiento, la cual puede ir más allá de las operaciones de almacenamiento y obtención de información. (Fig. 2.3.)

2. *Arquitectura de almacenamiento*. Sobre la manera en que se distribuyen las reponsabilidades entre los miembros del sistema de almacenamiento y sus interacciones. La Fig. 2.4 muestra esta taxonomía y la Fig. 2.5, su evolución.

Hablaremos con detalle de los sistemas de almacenamiento P2P, con especial atención a las estrategias de redundancia de información, en § 3 y § 3.4, respectivamente.

2.2.2. Medidas de desempeño

Además de la medidas de confiabilidad y disponibilidad revisadas en § 2.1.2 y § 2.1.1, el desempeño de los sistemas de almacenamiento puede analizarse a partir de las siguientes medidas [84]:

- *Ancho de banda.* Es la cantidad total de datos transferidos a través de un medio o sistema por unidad de tiempo.

- *Caudal (throughput).* Es la cantidad de trabajo realizado por un componente o sistema por unidad de tiempo. Caudal y ancho de banda son conceptos que suelen intercambiarse, no obstante, existen diferencias entre ellos: las operaciones reportadas por el caudal pueden tener distintos tamaños de datos.

- *Latencia.* Es el tiempo de espera entre el inicio de una petición y la primera reacción a ella.

- *Tiempo de respuesta.* Es el tiempo que le toma a una operación de almacenamiento dada terminar; desde que el usuario hace una solicitud y hasta que recibe la respuesta del sistema. Donde operación puede ser una de las siguientes: lectura, escritura, abrir, cerrar, buscar, o una combinación de los mismos.

- *Escalabilidad.* Es la habilidad del sistema de almacenamiento para crecer sin afectar negativamente el desempeño de su servicio.

- *Exceso de almacenamiento.* Es la cantidad de almacenamiento redundante en el sistema y necesaria para proveer un nivel de disponibilidad dado.

Figura 2.3: *Taxonomía conforme al propósito del sistema de almacenamiento distribuido* [67].

Figura 2.4: *Taxonomía conforme a la arquitectura del sistema de almacenamiento distribuido* [67].

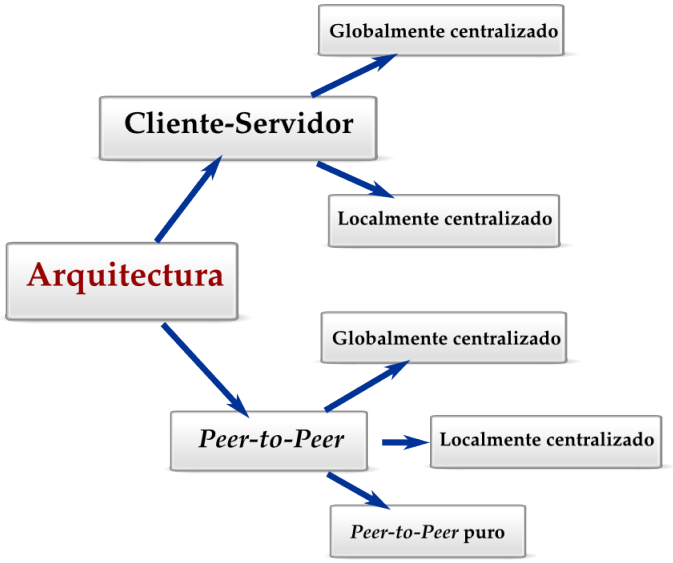
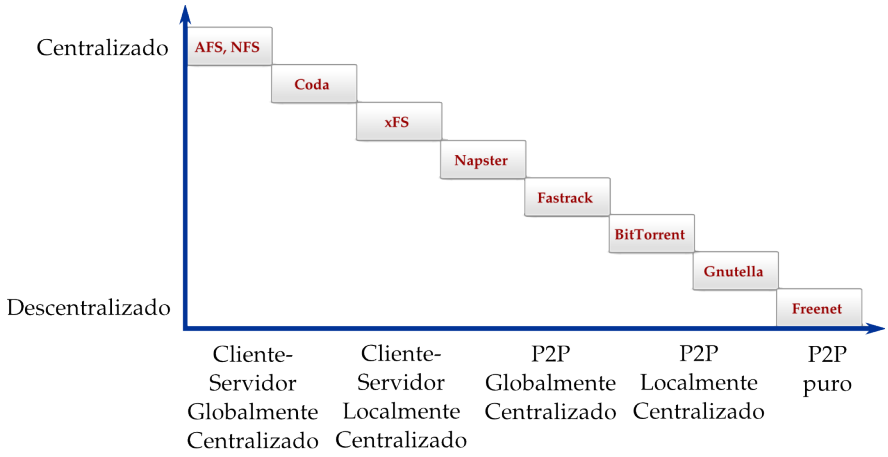


Figura 2.5: *Evolución de las arquitecturas para sistemas de almacenamiento distribuido* [67].



2.3. Tratamiento de la información

2.3.1. Disponibilidad

No se puede hablar formalmente de información sin mencionar el histórico trabajo desarrollado por Claude E. Shannon [81] a mediados del siglo XX, donde la información I se define en términos de la probabilidad ρ de que un mensaje dado sea recibido; esto es:

$$I = log\left(\frac{1}{\rho}\right) \equiv -log\,\rho \text{ [bits]}$$

Donde se aprecia que si ρ es pequeña entonces I puede tener un valor muy grande. Esto sigue de la noción de que el reporte de un evento improbable provee mayor información del que no lo es. De este modo, la información sólo depende de su probabilidad de ocurrencia y no de su contenido. Este concepto de información será el que tengamos en cuenta a lo largo de este trabajo de investigación; en particular, ahora que daremos significado a la disponibilidad de la información.

Si bien la disponibilidad es una medida de desempeño que definimos en § 2.1.2, es importante reiterar que ésta *únicamente* se refiere a la que ofrecen los sistemas de cómputo en general. En vista de esto, ¿a qué le llamamos *disponibilidad de la información*?, y ¿en qué se distingue de la *disponibilidad de los sistemas de cómputo*? Ésta nos habla de la probabilidad de que el servicio entregado por un sistema de cómputo dado opere adecuadamente en el instante solicitado; aquénlla, en cambio, la podemos pensar como la utilidad o el bien que un servicio entrega por medio de un sistema de cómputo.

En otras palabras, así como un sistema de cómputo puede servir ciclos de CPU, acceso a internet, espacio de almacenamiento, o energía eléctrica a través de un puerto USB, también es capaz de *servir información* a sus clientes.

De lo anterior se sigue que tiene sentido hablar de disponibilidad de la información sólo como un equivalente *cuantitativo* de la disponibilidad del sistema que la provea, justamente porque es éste quien es suceptible de fallar con cierta probabilidad, y no así la información por sí misma.

En el contexto particular de este trabajo de investigación, la disponibilidad de la información es equivalente a la de los sistemas de almacenamiento que la facilitan. Esto se podrá apreciar con mayor claridad en los experimentos realizados en la Parte II de este documento. Por otra parte, la disponibilidad de la información será mayor o menor en función de la redundancia que se haga de ella a lo largo de un sistema. ¿Cómo conseguir esa redundancia? En la siguiente sección responderemos esta pregunta.

2.3.2. Redundancia

El objetivo de esta sección es presentar los fundamentos teóricos detrás de las técnicas que se utilizan para aumentar el grado de disponibilidad de la información. Dichas técnicas tienen origen en la llamada teoría de códigos. Puesto que las técnicas de codificación son de gran relevancia en los sistemas de almacenamiento, en subsecciones siguientes profundizaremos en tres de ella que son fundamentales para nuestro trabajo: códigos de borrado, códigos de red, y las fuentes digitales.

Un código [59], en términos generales, se define como la asignación uno-a-uno entre un conjunto de mensajes por ser transmitidos y un conjunto de *palabras código* que son usados para transmitir esos mensajes; como en el ejemplo que se muestra en la Tabla 2.4.

La información, cuando se transmite a través de un canal de comunicación ruidoso —i.e., no confiable— puede corromperse en el camino hacia su destino. Shannon garantizó [81], sin embargo, que la información puede codificarse en la fuente —i.e., antes de la transmisión—, dando como resultado información distinta de la original que luego puede decodificarse en el destino con un grado de precisión especificado de antemano (Fig 2.6), siempre que la velocidad de transmisión no exceda la capacidad del canal.

Tabla 2.4: *Un código para cada mensaje.* [59].

Mensaje	Código
$\mathbf{m}_1$	101
$\mathbf{m}_2$	01
$\mathbf{m}_3$	110
$\mathbf{m}_4$	000

Ése es el problema fundamental de la teoría de códigos: determinar cuál es el mensaje enviado por un emisor —información original— a partir de la cadena de símbolos entregada al receptor. La información original puede *recuperarse* gracias a la información *redundante* generada en ella con un codificador.

Figura 2.6: *Transmisión de información por un canal de comunicación con errores.*

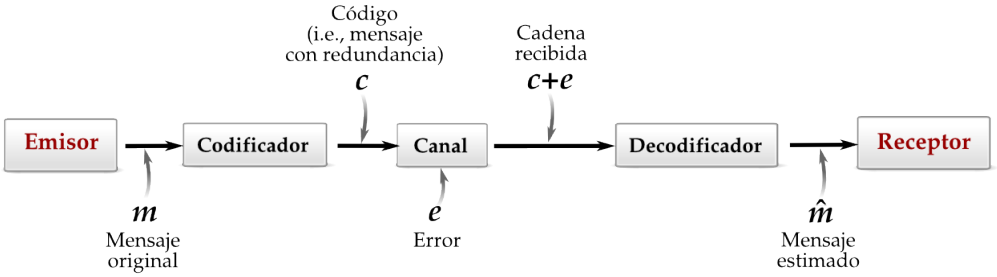
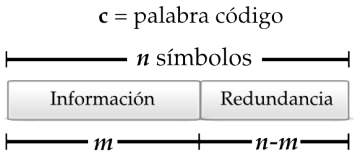


Figura 2.7: *Cadena devuelta por el codificador, i.e., codificación del original: información + redundancia.*



De los tipos de códigos existentes, los lineales son algunos de los más utilizados por su facilidad de codificación, decodificación y descripción. En ellos, los símbolos que representan la palabra código se suelen obtener de un conjunto $\mathbb{F}_p$, con p elementos, y el mensaje $\mathbf{m}$ y su codificación $\mathbf{c}$ (Figs. 2.6 y 2.7) son vectores en los espacios vectoriales $\mathbb{F}_p^m$ y $\mathbb{F}_p^n$, respectivamente. El conjunto $\mathbb{F}_p$ se conoce como *campo finito*.

Campos finitos

Se llama campo finito al conjunto de números enteros módulo p —mod p, de aquí en adelante— cerrado bajo las operaciones de suma, resta y multiplicación y división —siempre que el divisor sea distinto de cero—, denotado por $\mathbb{Z}_p$ [41, 76]. Esto es:

$$\mathbb{Z}_p = \frac{\mathbb{Z}}{p\,\mathbb{Z}} = \{\overline{0}, \overline{1}, \overline{2}, \ldots, \overline{p-1}\},$$

donde p es un número primo. Cada elemento del campo representa una clase, misma que es inducida por una relación de equivalencia. Se dice entonces que a está relacionado con b, $a \sim b$, si $b - a = c\,p$, donde $a, b, c, p \in \mathbb{Z}$. Asimismo, se acostumbra denotar la relación de equivalencia como:

$$a \equiv b \bmod p \iff b - a = c\ p$$

Por otra parte, para un número primo p, sea $\mathbb{F}_p = 0, 1, 2, \ldots, p-1$ y $\psi : \frac{\mathbb{Z}}{p\mathbb{Z}} \mapsto \mathbb{F}_p$, donde $\psi(\overline{a} = a)$ para $a = 0, 1, 2, \ldots, p-1$. $\mathbb{F}_p$ obtiene la estructura de un campo finito mediante el mapeo ψ, y es llamado *campo de Galois de orden* p. Dicho campo también suele denotarse como GF(p).

Hay tantos campos finitos como números primos. La elección del campo depende enteramente del problema a resolver, siendo las extensiones del campo $\mathbb{F}_2$, representadas como $\mathbb{F}_{2^b}$, las de mayor importancia para aplicaciones computacionales. Donde b es un número entero mayor o igual a cero.

Algoritmo de dispersión de información

El *algoritmo de dispersión de información* —IDA, por sus siglas en inglés— de M. O. Rabin [70] propone que para el problema de pérdida de información en un sistema de cómputo, se considere la opción de distribuir copias de ella en varios nodos en una red de computadoras. Los tres aspectos básicos que comprende al IDA son la *división*, *codificación* —o *dispersión*—, y *reconstrucción* de la información. A continuación detallamos cada uno de ellos.

- *División de la información.*

  Sea F un archivo formado por una cadena de N símbolos $s_{i,j}$, cada uno de los cuales se puede entender como elemento de un campo finito. Por ejemplo, si $s_{i,j}$ tiene 8 bits, entonces $s_{i,j} \in \mathbb{F}_{2^8}$. Otra posibilidad es considerar que $s_{i,j}$ es un entero tomado del intervalo $[0 \ldots p-1]$, donde p es un número primo, entonces F es una cadena de residuos mod p, es decir, una cadena de elementos de $\mathbb{F}_p$. Sin pérdida de generalidad, supondremos esta última posibilidad.

  Queremos *dispersar* F en n piezas tales que basten cualesquiera m de ellas, con $m < n$, para recuperar F. Escogemos un entero m de modo que $n = m + r$ satisface $n/m \leq 1 + r/m$, con $r/m > 0$ y r como el número de símbolos sumados a los originales durante la redundancia. Entonces F se divide en piezas de longitud m, de manera tal que

  $$F = (s_{1,1}, \ldots, s_{1,m}), (s_{2,1}, \ldots, s_{2,m}), \ldots, (s_{N/m,1}, \ldots, s_{N/m,m}).$$

  En notación matricial —para efecto de realizar operaciones—, esto es:

  $$\mathbf{F} = \begin{bmatrix} s_{1,1} & \cdots & s_{1,m} \\ s_{2,1} & \cdots & s_{2,m} \\ \vdots & \ddots & \vdots \\ s_{N/m,1} & \cdots & s_{N/m,m} \end{bmatrix} = \begin{bmatrix} \mathbf{f}_1 \\ \mathbf{f}_2 \\ \vdots \\ \mathbf{f}_{N/m} \end{bmatrix},$$

  Donde $\mathbf{F}$ es la representación matricial del archivo F; $s_{i,j} \in \mathbb{F}_p$ es un símbolo, y $\mathbf{f}_i$ una pieza original del archivo representado por $\mathbf{F}$.

- *Dispersión de la información.*

  Escogemos n vectores de coeficientes $\mathbf{a}_i = [a_{i,1} \quad a_{i,2} \quad \ldots \quad a_{i,m}]$, con $a_{i,j} \in \mathbb{F}_p$, para $i = 1 \ldots n$, donde:

  $$\mathbf{A} = \begin{bmatrix} \mathbf{a}_1 \\ \vdots \\ \mathbf{a}_n \end{bmatrix} = \begin{bmatrix} a_{1,1} & \cdots & a_{1,m} \\ \vdots & \ddots & \vdots \\ a_{n,1} & \cdots & a_{n,m} \end{bmatrix}$$

Se debe garantizar que cualesquiera m renglones son linealmente independientes aún cuando para muchas aplicaciones será suficiente asumir que, con alta probabilidad, un conjunto elegido aleatoriamente de m vectores $\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_m$ en $\mathbf{A}$ es linealmente independiente. Ahora bien, sea $\mathbf{D}$ la matriz que representa la *dispersión* de $\mathbf{F}$, definida como:

$$\mathbf{D} = \mathbf{A} \cdot \mathbf{F}^T, \tag{2.1}$$

es decir,

$$\mathbf{D} = \begin{bmatrix} a_{1,1} & \cdots & a_{1,m} \\ \vdots & \ddots & \vdots \\ a_{n,1} & \cdots & a_{n,m} \end{bmatrix} \cdot \begin{bmatrix} s_{1,1} & \cdots & s_{N/m,1} \\ \vdots & \ddots & \vdots \\ s_{1,m} & \cdots & s_{N/m,m} \end{bmatrix}$$

$$= \begin{bmatrix} \mathbf{a}_1 \mathbf{f}_1^T & \cdots & \mathbf{a}_1 \mathbf{f}_{N/m}^T \\ \vdots & & \vdots \\ \mathbf{a}_n \mathbf{f}_1^T & \cdots & \mathbf{a}_n \mathbf{f}_{N/m}^T \end{bmatrix}$$

$$= \begin{bmatrix} d_{1,1} & \cdots & d_{1,N/m} \\ \vdots & & \vdots \\ d_{n,1} & \cdots & d_{n,N/m} \end{bmatrix} = \begin{bmatrix} \mathbf{d}_1 \\ \vdots \\ \mathbf{d}_n \end{bmatrix}.$$

Entonces, para $j = 1 \ldots N/m$ y $l = 1 \ldots m$,

$$d_{i,j} = \mathbf{a}_i \mathbf{f}_j^T = \sum_{l=1}^{m} a_{i,l} s_{j,l},$$

con $d_{i,j} \in \mathbb{F}_p$.

Para efectos de este trabajo, *dispersión* y *codificación* de información se tratarán como conceptos equivalentes. Asimismo, un *disperso* no es sino una pieza de archivo codificada para su posterior almacenamiento y recuperación.

- *Reconstrucción de la información*.

  Si se recuperan m dispersos $\mathbf{d}_1 \ldots \mathbf{d}_m$, y sus m vectores de coeficientes asociados $\mathbf{a}_1 \ldots \mathbf{a}_m$, podemos reconstruir la pieza del archivo F representada por el vector $\mathbf{f}_j^T$ del siguiente modo. Sea $\hat{\mathbf{A}} = (a_{i,j})_{1 \leq i,j \leq m}$ la matriz $m \times m$, que es una versión parcial de $\mathbf{A}$, y cuya i-ésima fila es $\mathbf{a}_i$, tal que:

$$\hat{\mathbf{A}} \cdot \mathbf{f}_j^T = \begin{bmatrix} d_{1,j} \\ \vdots \\ d_{m,j} \end{bmatrix},$$

y de aquí que

$$\mathbf{f}_j^T = \hat{\mathbf{A}}^{-1} \cdot \begin{bmatrix} d_{1,j} \\ \vdots \\ d_{m,j} \end{bmatrix},$$

es decir,

$$\begin{bmatrix} s_{j,1} \\ \vdots \\ s_{j,m} \end{bmatrix} = \begin{bmatrix} a_{1,1} & \cdots & a_{1,m} \\ \vdots & \ddots & \vdots \\ a_{m,1} & \cdots & a_{m,m} \end{bmatrix}^{-1} \cdot \begin{bmatrix} d_{1,j} \\ \vdots \\ d_{m,j} \end{bmatrix}$$

Si el i-ésimo renglón de $\hat{\mathbf{A}}^{-1}$ es el vector $[\alpha_{i,1} \quad \dots \quad \alpha_{i,m}]$, entonces

$$s_{j,l} = \sum_{i=1}^{m} \alpha_{l,i} d_{i,j}, \tag{2.2}$$

para $l = 1 \dots m$ y $j = 1 \dots N/m$.

En resumen, IDA cuenta con las siguientes características:

1. IDA es un método para transformar un archivo F en n piezas, denominados *dispersos*.
2. El archivo F, de tamaño $|F|$ símbolos, puede reconstruirse a partir de cualesquiera m de n piezas. Cada pieza es de tamaño $|F|/m$. Así que el número total de caracteres que componen el archivo transformado es $\frac{n}{m}|F|$.
3. Es posible escoger n de modo que $k = \frac{n}{m} \approx 1$; donde k es el llamado *factor de estiramiento*. De esta manera, IDA es eficiente en el uso de almacenamiento; asimismo, la codificación, distribución y reconstrucción de la información, también son eficientes computacionalmente (Tabla 2.6).

IDA pertenece al conjunto de los códigos de corrección de errores, en los que se añaden bits extras a un mensaje, creando un bloque (Fig 2.7), de modo que después de la ocurrencia de cualesquiera $n - m$ errores dentro del bloque, el mensaje aún puede ser reconstruido. Esos $n - m$ errores puede localizarse en cualquier lugar dentro del bloque, y producidos por un bit cambiado o eliminado. En particular, IDA es una técnica que utiliza códigos de borrado del tipo conocido como *Maximum-Distance Separable* (MDS) [57].

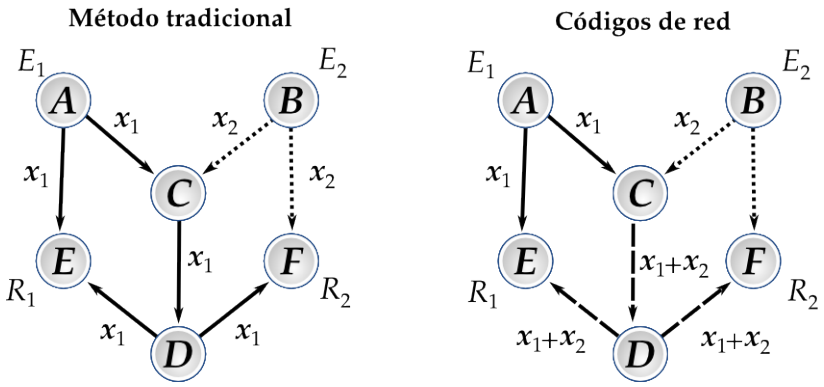
Supongamos un archivo F de 10000 símbolos, el cual se pretende codificar y distribuir en $n = 14$ piezas, de las cuales $m = 10$ serán suficientes para reconstruir F, cada una de tamaño $|F|/10 = 1000$, implicando un exceso de información del 40 %. El perder 4 piezas —bien identificadas— es equivalente a la pérdida de 4 caracteres en las mismas posiciones identificadas dentro de cada bloque. Los parámetros n y m pueden elegirse como sea necesario. Esta configuración también conlleva un exceso de 40 % en el número de caracteres transmitidos. Sin embargo, transmitir la i-ésima pieza codificada de F, f_i, de A a B a lo largo de $n = 14$ canales también supone un beneficio extra: una mejor distribución de la carga de la red.

Expuesto lo anterior, en adelante tomaremos IDA como referencia para hablar de la distribución de información a lo largo de sistemas de almacenamiento distribuido que utilicen técnicas de similares a los códigos correctores de errores; en particular, para códigos de borrado tipo MDS.

Códigos de red

Los códigos de red [3] conforman un área relativamente nueva de investigación que puede ofrecen importantes aplicaciones en sistemas de almacenamiento distribuido [32, 24]. Con códigos de red, los nodos intermedios entre la fuente y el destino pueden reexpedir paquetes que son combinaciones lineales de la información previamente recibida; es decir, donde el proceso de codificación y decodificación se realiza durante el camino y no sólo en los extremos. En principio, los más importantes beneficios de usar códigos de red son la potencial mejora del caudal —*throughput*— de información, así como el alto grado de robustez

Figura 2.8: *Los nodos emisores E_1 y E_2 envían información con* multicast *a los nodos receptores R_1 y R_2, cada uno. Todos los enlaces tienen capacidad igual a 1. Con códigos de red, al combinar la información en el nodo C con operaciones XOR, se logran tasas de transmisión de 1 por cada emisor: como si usaran la red sólo para ellos mismos. Sin códigos de red, la tasa de transmisión es de* 1/2 *hacia cada receptor* [29].



y adaptabilidad del sistema resultante [29]. La Fig. 2.8 ilustra una situación en la que se realiza codificación en red.

En los códigos de red se presentan cuatro situaciones básicas:

- *División de la información*.

  Tengamos por caso un archivo F compuesto por N símbolos $s_{i,j} \in \mathbb{F}_p$, dividido en piezas $\mathbf{f}_i$ con m símbolos cada una, con $i = 1 \ldots N/m$ y $j = 1 \ldots m$. De este modo

$$\mathbf{F} = \begin{bmatrix} s_{1,1} & \cdots & s_{1,m} \\ \vdots & \ddots & \vdots \\ s_{N/m,1} & \cdots & s_{N/m,m} \end{bmatrix} = \begin{bmatrix} \mathbf{f}_1 \\ \vdots \\ \mathbf{f}_{N/m} \end{bmatrix},$$

  Si usamos una codificación lineal, tendremos *códigos de red lineales* donde las piezas codificadas, o dispersos, $\mathbf{d}$ son combinaciones lineales de las piezas originales, cuya aritmética se realiza sobre un campo $\mathbb{F}_p$. Con este tipo de codificación la longitud de los dispersos $\mathbf{d}$ equivale a la de las piezas originales $\mathbf{f}$, es decir, $|\mathbf{d}| = |\mathbf{f}_i| = m$ símbolos.

- *Dispersión de la información*.

  Si un nodo recibe un número N/m de piezas de archivo $\mathbf{f}_1 \ldots \mathbf{f}_{N/m}$ que son generadas por una o más fuentes; con estas piezas y un vector de coeficientes $\mathbf{a} = [a_1 \ \cdots \ a_{N/m}]$, con $a_i \in \mathbb{F}_p$, se crean *dispersos*

de red $\mathbf{d}$ tales que

$$\begin{aligned}
\mathbf{d} &= \mathbf{a}\cdot\mathbf{F} \\
&= \begin{bmatrix} a_1 & \cdots & a_{N/m} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{f}_1 \\ \vdots \\ \mathbf{f}_{N/m} \end{bmatrix} \\
&= \sum_{i=1}^{N/m} a_i \mathbf{f}_i \\
&= a_1\mathbf{f}_1 + \cdots + a_{N/m}\mathbf{f}_{N/m} \\
&= a_1 \cdot [\, s_{1,1} \quad \cdots \quad s_{1,m}] + \cdots + a_{N/m} \cdot [\, s_{N/m,1} \quad \cdots \quad s_{N/m,m}] \\
&= \begin{bmatrix} \sum_{i=1}^{N/m} a_i s_{i,1} & \cdots & \sum_{i=1}^{N/m} a_i s_{i,m} \end{bmatrix}.
\end{aligned}$$

De este modo,

$$\mathbf{d} = [d_1 \quad \cdots \quad d_m],$$

con

$$d_j = \sum_{i=1}^{N/m} a_i s_{i,j}.$$

- *Recombinación de la información*.

  Después de la codificación, se envían paquetes a otros nodos en la red, cada uno de los cuales está conformado por un vector de coeficientes $\mathbf{a}_i = [a_{i,1} \quad \cdots \quad a_{i,N/m}]$ y el i-ésimo disperso de red $\mathbf{d}_i$. Para la creación de un nuevo disperso de red $\mathbf{d}_{n+1}$ y su vector de coeficientes $\mathbf{a}_{n+1}$, en algún nodo de la red, se hace una *recombinación* de los paquetes recibidos ahí.

  Sean $(\mathbf{a}_1, \mathbf{d}_1)\ldots(\mathbf{a}_n, \mathbf{d}_n)$ un conjunto de n paquetes recibidos en algún nodo de la red; un nuevo disperso $\mathbf{d}_{n+1}$ se obtiene con

$$\begin{aligned}
\mathbf{d}_{n+1} &= \sum_{i=1}^{n} a_i' \mathbf{d}_i \\
&= a_1'\mathbf{d}_1 + \cdots + a_n'\mathbf{d}_n \\
&= a_1' \cdot [d_{1,1} \cdots d_{1,m}] + \cdots + a_n' \cdot [d_{n,1} \cdots d_{n,m}] \\
&= \begin{bmatrix} \sum_{i=1}^{n} a_i' d_{i,1} & \cdots & \sum_{i=1}^{n} a_i' d_{i,m} \end{bmatrix},
\end{aligned}$$

  donde

$$a_{n+1,j} = \sum_{i=1}^{n} a_i'\, a_{i,j},$$

es el j-ésimo elemento del vector de coeficientes $\mathbf{a}_{n+1}$ asociado al disperso $\mathbf{d}_{n+1}$, resultado de una recombinación lineal de los n coeficientes ubicados en la j-ésima posición, con $j = 1 \dots m$, y los coeficientes $a'_i \in \mathbb{F}_p$ de un vector $\mathbf{a}'$ creado en el nodo donde se realiza la recodificación.

- *Reconstrucción de la información*.

  Para decodificar un archivo **F**, realizamos el siguiente procedimiento. Supongamos que en un nodo de la red recibe el conjunto de paquetes $(\mathbf{a}_1, \mathbf{d}_1) \dots (\mathbf{a}_n, \mathbf{d}_n)$, con $n \geq N/m$, y sea $\hat{\mathbf{A}}$ la matriz de coeficientes cuyos renglones son N/m vectores entre $\mathbf{a}_1, \cdots, \mathbf{a}_n$, entonces tomamos cualesquiera N/m de n paquetes para resolver el sistema

$$\begin{bmatrix} \mathbf{d}_1 \\ \vdots \\ \mathbf{d}_{N/m} \end{bmatrix} = \hat{\mathbf{A}} \cdot \begin{bmatrix} \mathbf{f}_1 \\ \vdots \\ \mathbf{f}_{N/m} \end{bmatrix},$$

  para las piezas originales de **F**; entonces

$$\begin{bmatrix} \mathbf{f}_1 \\ \vdots \\ \mathbf{f}_{N/m} \end{bmatrix} = \hat{\mathbf{A}}^{-1} \cdot \begin{bmatrix} \mathbf{d}_1 \\ \vdots \\ \mathbf{d}_{N/m} \end{bmatrix}$$
$$= \begin{bmatrix} a_{1,1} & \cdots & a_{1,N/m} \\ \vdots & \ddots & \vdots \\ a_{N/m,1} & \cdots & a_{N/m,N/m} \end{bmatrix}^{-1} \cdot \begin{bmatrix} \mathbf{d}_1 \\ \vdots \\ \mathbf{d}_{N/m} \end{bmatrix}$$

  Si el vector $[\alpha_{i,1} \quad \dots \quad \alpha_{i,N/m}]$ es el i-ésimo renglón de $\hat{\mathbf{A}}^{-1}$, con $\alpha_{i,j} \in \mathbb{F}_p$, entonces

$$\mathbf{f}_j = \sum_{i=1}^{N/m} \alpha_{j,i} \, \mathbf{d}_i, \tag{2.3}$$

  y

$$s_{j,l} = \sum_{i=1}^{N/m} \alpha_{j,i} \, d_{i,l}, \tag{2.4}$$

  para $l = 1 \dots m$ y $j = 1 \dots N/m$.

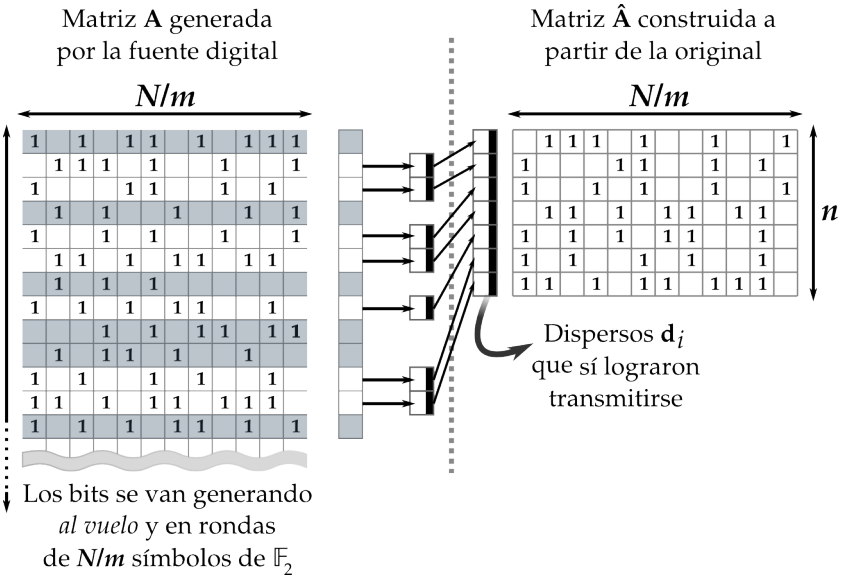
Fuentes digitales

Las fuentes digitales son una clase de ćodigos correctores de errores —también conocidos como *códigos sin tasa*— que para bloques grandes información son varios órdenes de magnitud más rápidos que los códigos de borrado convencionales.

Las fuentes digitales suelen explicarse mediante la metáfora de una fuente de agua, que es una productora inacabable de gotas, de las que sólo basta recabar $\mathcal{M}$ de ellas, *cualesquiera*, para llenar un recipiente con una capacidad dada.

En nuestro contexto, lo anterior significa que un nodo puede descargar, de una o más fuentes, paquetes codificados de cierta información, hasta tener bastantes para reconstruirla mediante algún proceso de decodificación, y sin importar qué paquetes fueron obtenidos. Idealmente, una fuente digital cuenta con las siguientes dos propiedades [16]:

Figura 2.9: *Cuando los dispersos son transmitidos, algunos no son recibidos —mostrados con sombreado—, y cada uno corresponde a un renglón de la matriz* **A**. *Del lado de quien requiera recuperar información, los dispersos se pueden realinear para formar una nueva matriz* $\hat{\mathbf{A}}$, *como se muestra en la parte derecha del diagrama* [49].



1. Una fuente digital debe ser capaz de generar un número *ilimitado* de dispersos a partir de las piezas originales de información.

2. La recuperación —i.e., decodificación— de la información original debería lograrse a partir de *cualesquiera* $\mathcal{M}$ dispersos recibidos de una o más fuentes digitales. Además, el proceso de reconstrucción de la información original debe realizarse con una complejidad lineal $O(\mathcal{M})$.

En la práctica, los códigos sin tasa se dispersan como los códigos de borrado convencionales, excepto por el hecho de que existe un ligero exceso ϵ en la cantidad $\mathcal{M}$ de dispersos requeridos para recuperar la información original. Dicha cantidad, $\mathcal{M}+\epsilon$, puede determinarse *al vuelo* al tiempo que también se van creando los dispersos necesarios.

Las fuentes digitales también pueden construirse con códigos tradicionales, aunque con algunas limitaciones. Por ejemplo, para construir una fuente digital con códigos de borrado como el IDA, se convierte el número de paquetes codificados n en múltiplo del número de dispersos m que requiere para la reconstrucción de la información, y entonces se procede a transmitir desde una fuente de datos los n dispersos en una cantidad de ciclos delimitada por el múltiplo elegido. Sin embargo, la principal limitación de utilizar estos códigos en fuentes digitales es la pobre eficiencia computacional de las operaciones de dispersión y recuperación de la información —codificación y decodificación, respectivamente—, normalmente de orden cuadrático.

Otro aspecto que limita el uso de códigos de borrado como fuentes digitales es que aquellos requieren la estimación de una probabilidad ρ de transmitir la información sin errores. Dicha estimación le ofrece al diseñador del sistema la oportunidad de elegir una tasa $\frac{m}{n}$ para el código antes de la transmisión de la información. La elección de una tasa adecuada puede evitar que se reciban menos de m dispersos para la reconstrucción de la información en sistemas donde las probabilidades de falla $1-\rho$ sean más grandes de lo esperado. Más aún, los códigos de borrado tipo MDS, como IDA, no son capaces de crear más dispersos al vuelo que los n creados originalmente.

Para comprender el funcionamiento de las fuentes digitales, usaremos el ejemplo propuesto por [49]:

una fuente digital lineal aleatoria (Fig. 2.9). Es la fuente digital más sencilla de todas, y sin embargo útil por ser representativa del tema de nuestra exposición. Dicho lo anterior, nuestra fuente digital consta de las siguientes etapas:

- *División de la información*.

  Para esto consideremos la división de un archivo F que consta de N símbolos, y dividido en piezas $\mathbf{f}_i$ con m símbolos $s_{i,j}$ cada una. A su vez, cada símbolo $s_{i,j} \in \mathbb{F}_2$, específicamente.

$$\mathbf{F} = \begin{bmatrix} s_{1,1} & \cdots & s_{1,m} \\ \vdots & \ddots & \vdots \\ s_{N/m,1} & \cdots & s_{N/m,m} \end{bmatrix} = \begin{bmatrix} \mathbf{f}_1 \\ \vdots \\ \mathbf{f}_{N/m} \end{bmatrix}$$

- *Dispersión de la información.*

  Existe una matriz de coeficientes $\mathbf{A} = (a_{i,j})_{i\geq 1,\ 1\leq j\leq N/m}$, donde $a_{i,j}$ obtiene su valor de un generador aleatorio de bits —i.e., se crean símbolos $a_{i,j} \in \mathbb{F}_2$—. Los bits se generan en rondas de N/m bits para cada vector de coeficientes $\mathbf{a}_i = [a_{i,1} \quad \cdots \quad a_{i,N/m}]$.

$$\mathbf{A} = \begin{bmatrix} a_{1,1} & \cdots & a_{1,N/m} \\ \vdots & \ddots & \vdots \\ a_{i,1} & \cdots & a_{i,N/m} \\ \vdots & \ddots & \vdots \end{bmatrix} = \begin{bmatrix} \mathbf{a}_1 \\ \vdots \\ \mathbf{a}_i \\ \vdots \end{bmatrix}$$

  Así pues, el i-ésimo disperso $\mathbf{d}_i$ creado en la fuente digital se define como

$$\begin{aligned} \mathbf{d}_i &= \mathbf{a}_i \cdot \mathbf{F} \\ &= \mathbf{a}_i \cdot \begin{bmatrix} \mathbf{f}_1 \\ \vdots \\ \mathbf{f}_{N/m} \end{bmatrix} \\ &= [a_{i,1} \quad \cdots \quad a_{i,N/m}] \cdot \begin{bmatrix} \mathbf{f}_1 \\ \vdots \\ \mathbf{f}_{N/m} \end{bmatrix} \\ &= a_{i,1}\mathbf{f}_1 \oplus \cdots \oplus a_{i,N/m}\mathbf{f}_{N/m} \\ &= \sum_{j=1}^{N/m} a_{i,j}\mathbf{f}_j. \end{aligned}$$

  De modo que

$$d_{i,l} = \sum_{j=1}^{N/m} a_{i,j} s_{j,l},$$

  donde $d_{i,l} \in \mathbb{F}_2$.

- *Reconstrucción de la información.*

  Por el otro lado, en el receptor de la información, que espera la llegada de al menos n dispersos, cualesquiera, ¿cuál es la probabilidad de que con esos n dispersos pueda reconstruir la información original? En este caso, podemos asumir que el receptor conoce una parte de la matriz $\mathbf{A}$ asociada a los dispersos, y que fue creada con un generador de números aleatorios similar al utilizado en el receptor. Denotamos la matriz que el receptor recupera como $\hat{\mathbf{A}} = (a_{i,j})_{1\leq i\leq n,\ 1\leq j\leq N/m}$.

  Así pues, si $n < N/m$, el receptor no tendrá suficiente información para recuperar el archivo original. Si $n = N/m$, es posible que el archivo pueda recuperarse, pero ¿con qué probabilidad? Más específicamente, para este caso, si se tiene $\hat{\mathbf{A}}$, de dimensión $n \times n$, debemos calcular cuál es la probabilidad que de sea invertible. En [49] se calcula que esa probabilidad es de 0.289 para cualquier $N/m > 10$. Entonces será posible recuperar la pieza $\mathbf{f}_i$ de la siguiente manera. Si los dispersos se codificaron de

modo que

$$\begin{bmatrix} \mathbf{d}_1 \\ \vdots \\ \mathbf{d}_{N/m} \end{bmatrix} = \begin{bmatrix} a_{1,1} & \cdots & a_{1,N/m} \\ \vdots & \ddots & \vdots \\ a_{N/m,1} & \cdots & a_{N/m,N/m} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{f}_1 \\ \vdots \\ \mathbf{f}_{N/m} \end{bmatrix},$$

y para la decodificación se utiliza la matriz $\hat{\mathbf{A}}$, cuya matriz inversa es $\hat{\mathbf{A}}^{-1}$ en la que el i-ésimo renglón es $[\alpha_{i,1} \quad \dots \quad \alpha_{i,N/m}]$, entonces

$$\begin{bmatrix} \mathbf{f}_1 \\ \vdots \\ \mathbf{f}_{N/m} \end{bmatrix} = \begin{bmatrix} \alpha_{1,1} & \cdots & \alpha_{1,N/m} \\ \vdots & \ddots & \vdots \\ \alpha_{N/m,1} & \cdots & \alpha_{N/m,N/m} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{d}_1 \\ \vdots \\ \mathbf{d}_{N/m} \end{bmatrix},$$

de donde

$$\begin{aligned} \mathbf{f}_j &= \alpha_{j,1}\,\mathbf{d}_1 \oplus \cdots \oplus \alpha_{j,N/m}\,\mathbf{d}_{N/m} \\ &= \sum_{i=1}^{N/m} \alpha_{j,i}\mathbf{d}_i, \end{aligned}$$

y

$$s_{j,l} = \sum_{i=1}^{N/m} \alpha_{j,i} d_{i,l}.$$

En otro caso, si n es ligeramente mayor que N/m, sea $n = \frac{N}{m} + \epsilon$, donde ϵ es un número pequeño que representa un exceso de paquetes recibidos. ¿Cuál es la probabilidad de que la matriz $\hat{\mathbf{A}}$ de $n \times \frac{N}{m}$ en el receptor contenga una matriz invertible de dimension $\frac{N}{m} \times \frac{N}{m}$? Sea $1 - \delta$ la probabilidad de encontrar esa matriz; entonces la probabilidad de fallo δ está delimitada por

$$\delta(\epsilon) \leq 2^{-\epsilon}.$$

Así que el número de dispersos requeridos para tener una probabilidad de éxito $1 - \delta$ en la reconstrucción de la información original es $\approx \frac{N}{m} + log_2 \frac{1}{\delta}$.

Los códigos LT [48], Raptor [82] y Online [51], son códigos sin tasa de mucho mejor calidad que los lineales aleatorios, tanto en la complejidad computacional de los procesos de dispersión y recuperación de la información, como en la probabilidades de éxito para recuperar información.

Entre las potenciales aplicaciones de los códigos sin tasa están la descarga de datos de forma paralela, TCP de uno-a-muchos, *streaming* de vídeo, y la transmisión sobre redes superpuestas. Los códigos sin tasa resuelven de manera natural problemas típicos de almacenamiento distribuido como la retransmisión debida a pérdidas, porque cada paquete codificado es valioso para reconstruir la información original.

Resumen y comparación de esquemas de codificación

En este apartado mostramos una serie de tablas (Tablas 2.5, 2.6, 2.7 y 2.8) que resumen lo dicho sobre los códigos de borrado IDA, códigos de red y fuentes digitales. Para las cuales tomamos como referencia un archivo definido de la siguiente manera:

Sea un archivo F con N símbolos que pertenecen a un campo finito $\mathbb{F}_p$. El archivo se divide en piezas de tamaño m símbolos; de este modo, F queda representado por la matriz $\mathbf{F}$ y sus piezas por los vectores $\mathbf{f}_i$, cuyos símbolos se denotan como $s_{i,j}$, para $i = 1 \ldots N/m$ y $j = 1 \ldots m$. Cada pieza original se codifica, mediante vectores de coeficientes o símbolos de $\mathbb{F}_p$, para crear dispersos $\mathbf{d}_i$, con elementos $d_{i,j}$.

$$\mathbf{F} = \begin{bmatrix} s_{1,1} & \cdots & s_{1,m} \\ s_{2,1} & \cdots & s_{2,m} \\ \vdots & \ddots & \vdots \\ s_{N/m,1} & \cdots & s_{N/m,m} \end{bmatrix} = \begin{bmatrix} \mathbf{f}_1 \\ \mathbf{f}_2 \\ \vdots \\ \mathbf{f}_{N/m} \end{bmatrix}.$$

Tabla 2.5: *Comparación de elementos de los códigos.*

Codificación	$\lvert\mathbf{f}_i\rvert$ [*símbolos*]	$\lvert\mathbf{a}_i\rvert$ [*símbolos*]	$\lvert\mathbf{d}_i\rvert$ [*símbolos*]	Dispersos creados a partir de $\mathbf{F}$	$\lvert\hat{\mathbf{A}}\rvert$ [*símbolos*]	Campo finito
Códigos IDA	m	m	N/m	n (fijo)	m^2	$\mathbb{F}_p$
Códigos de red	m	N/m	m	$n \to \infty$	$(N/m)^2$	$\mathbb{F}_p$
Fuentes digitales	m	N/m	m	$n \to \infty$	$(N/m)^2$	$\mathbb{F}_2$

Tabla 2.6: *Costos computacionales de codificación y decodificación.*

Código	**Costo de crear un símbolo codificado**	**Costo de recuperar N símbolos $s_{i,j}$**
Códigos IDA [70]	$O(m)$	$O(mN + m^3)$
Códigos de red [52]	$O(N/m)$	$O(\,(N^2/m) + (N/m)^3\,)$
Fuente digital (Raptor) [82]	$O(\,\log(1/\epsilon)\,)$	$O(\,N\log(1/\epsilon)\,)$

Tabla 2.7: *Resumen de la etapa de dispersión.*

Tipo de codificación	Vector de coeficientes	Dispersos
Códigos IDA	Para $i = 1 \dots n$, y $n > m$ $$\mathbf{a}_i = \begin{bmatrix} a_{i,1} & \cdots & a_{i,m} \end{bmatrix}.$$	Para $i = 1 \dots n$ $$\mathbf{d}_i = \begin{bmatrix} \mathbf{a}_i \mathbf{f}_1^T & \cdots & \mathbf{a}_i \mathbf{f}_{N/m}^T \end{bmatrix},$$ donde $$d_{i,j} = \mathbf{a}_i \mathbf{f}_j^T = \sum_{l=1}^{m} a_{i,l} s_{j,l},$$ y $j = 1 \dots N/m$.
Códigos de red	■ Con N/m piezas originales. $$\mathbf{a} = \begin{bmatrix} a_1 & \cdots & a_{N/m} \end{bmatrix}$$ ■ Recombinación con n dispersos de red. Existe un vector con coeficientes a'_i, para $i = 1 \dots N/m$, tales que $$a_{n+1,j} = \sum_{i=1}^{n} a'_i \, a_{i,j},$$ es el j-ésimo coeficiente del vector $\mathbf{a}_{n+1}$.	■ Con N/m piezas originales. $$\mathbf{d} = \mathbf{a} \cdot \mathbf{F} = \sum_{i=1}^{N/m} a_i \mathbf{f}_i,$$ y, para $j = 1 \dots m$ $$d_j = \sum_{i=1}^{N/m} a_i s_{i,j}.$$ ■ Recombinación con n dispersos de red. $$\mathbf{d}_{n+1} = \sum_{i=1}^{n} a'_i \mathbf{d}_i,$$ y, para $j = 1 \dots m$ $$d_{n+1,j} = \sum_{i=1}^{n} a'_i d_{i,j}$$
Fuentes digitales	Para toda $i \geq 1$ $$\mathbf{a}_i = \begin{bmatrix} a_{i,1} & \cdots & a_{i,N/m} \end{bmatrix}.$$	Para toda $i \geq 1$ $$\mathbf{d}_i = \mathbf{a}_i \cdot \mathbf{F} = \sum_{j=1}^{N/m} a_{i,j} \mathbf{f}_j,$$ y, para $l = 1 \dots m$ $$d_{i,l} = \sum_{j=1}^{N/m} a_{i,j} s_{j,l}.$$

Tabla 2.8: *Resumen de la etapa de reconstrucción de la información.*

Tipo de codificación	Matriz de coeficientes	Dispersos
Códigos IDA	Para cualesquiera m de n vectores de coeficientes linealmente independientes. $$\hat{\mathbf{A}} = \begin{bmatrix} a_{1,1} & \cdots & a_{1,m} \\ \vdots & \ddots & \vdots \\ a_{m,1} & \cdots & a_{m,m} \end{bmatrix},$$ y el i-ésimo renglón de $\hat{\mathbf{A}}^{-1}$ es $$\begin{bmatrix} \alpha_{i,1} & \cdots & \alpha_{i,m} \end{bmatrix}.$$	Para $l = 1 \ldots m$ y $j = 1 \cdots N/m$ $$\mathbf{f}_j^T = \hat{\mathbf{A}}^{-1} \cdot \begin{bmatrix} d_{1,j} \\ \vdots \\ d_{m,j} \end{bmatrix},$$ y $$s_{j,l} = \sum_{i=1}^{m} \alpha_{l,i} d_{i,j}.$$
Códigos de red	Para cualesquiera N/m de n vectores de coeficientes linealmente independientes. $$\hat{\mathbf{A}} = \begin{bmatrix} a_{1,1} & \cdots & a_{1,N/m} \\ \vdots & \ddots & \vdots \\ a_{N/m,1} & \cdots & a_{N/m,N/m} \end{bmatrix}$$ y el i-ésimo renglón de $\hat{\mathbf{A}}^{-1}$ es $$\begin{bmatrix} \alpha_{i,1} & \cdots & \alpha_{i,N/m} \end{bmatrix}.$$	Para $l = 1 \ldots m$ y $j = 1 \ldots N/m$ $$\mathbf{f}_j = \sum_{i=1}^{N/m} \alpha_{j,i}\, \mathbf{d}_i,$$ y $$s_{j,l} = \sum_{i=1}^{N/m} \alpha_{j,i}\, d_{i,l}.$$
Fuentes digitales	Para cualesquiera N/m de $n = \frac{N}{m} + \epsilon$ vectores de coeficientes linealmente independientes, con probabilidad $\delta(\epsilon)$. $$\hat{\mathbf{A}} = \begin{bmatrix} a_{1,1} & \cdots & a_{1,N/m} \\ \vdots & \ddots & \vdots \\ a_{N/m,1} & \cdots & a_{N/m,N/m} \end{bmatrix}$$ y el i-ésimo renglón de $\hat{\mathbf{A}}^{-1}$ es $$\begin{bmatrix} \alpha_{i,1} & \cdots & \alpha_{i,N/m} \end{bmatrix}.$$	Para $l = 1 \ldots m$ y $j = 1 \ldots N/m$ $$\mathbf{f}_j = \sum_{i=1}^{N/m} \alpha_{j,i}\, \mathbf{d}_i,$$ y $$s_{j,l} = \sum_{i=1}^{N/m} \alpha_{j,i} d_{i,l}.$$

2.3.3. Modelo de flujo de información

Para entender los límites que un sistema de almacenamiento distribuido tiene en términos de ancho de banda y almacenamiento, en lo que sigue explicamos un modelo de flujo de información definido inicialmente en [23]; antes definimos una red de flujo y el problema del flujo máximo para dicho tipo de redes.

Flujo máximo

Las redes de flujo se utilizan para modelar distintos tipos de materiales de flujo en tuberías, líneas de ensamblaje, redes eléctricas, redes de comunicación, entre otros. Cada arista dirigida en la red de flujo puede pensarse como un conducto por el que se transporta un material. Se dice que el problema más simple que concierne a las redes de flujo es el del *flujo máximo* [20]. Esto es: ¿cuál es la tasa más grande a la que un material puede transportarse de la fuente hasta el depósito sin violar los límites de capacidad de cada camino posible? Existen algoritmos eficientes para resolver este problema, e.g., el método de Ford y Fulkerson [20].

Formalmente, una red de flujo $\mathcal{G} = (V, E)$ es un grafo dirigido en el que cada arista $(u, v) \in E$ tiene una capacidad $c(u, v) \geq 0$. Si $(u, v) \notin E$, asumimos que $c(u, v) = 0$. Hay dos vértices en $\mathcal{G}$ que se distinguen de los demás: una fuente s y un sumidero t. Donde, para cada vértice $v \in V$, existe un camino $s \rightsquigarrow v \rightsquigarrow t$; por lo tanto el grafo está conectado, y $|E| \geq |V| - 1$. Se asume que no hay aristas dirigidas hacia s ni con origen en t.

Sea $\mathcal{G} = (V, E)$ una red de flujo, con una función de capacidad c. Sea s la fuente de la red, y sea t el sumidero. Un *flujo* en $\mathcal{G}$ es una función de valor real $f : V \times V \to \mathbb{R}$ que satisface las siguientes tres propiedades:

1. *Restricción de capacidad*: Para toda arista $(u, v) \in V$, requerimos $f(u, v) \leq c(u, v)$
2. *Simetría de sesgo*: Para toda arista $(u, v) \in V$, requerimos $f(u, v) = -f(v, u)$.
3. *Conservación de flujo*: Para todo vértice $u \in V - \{s, t\}$, requerimos $\sum_{v \in V} f(u, v) = 0$

La cantidad $f(u, v)$ es el *flujo de red* del vértice u al vértice v y puede ser positiva o negativa. El *valor* de un flujo f se define como

$$|f| = \sum_{v \in V} f(s, v),$$

esto es, el flujo de red total desde la fuente [3]. Así pues, en el problema del flujo máximo, tenemos una red de flujo $\mathcal{G}$ con una fuente s y un sumidero t, y lo que deseamos es encontrar un flujo $|f|$ de valor máximo de s a t. Esto nos da pie para enunciar el llamado teorema *flujo máximo – corte mínimo*.

Teorema 2.1 *Si f es un flujo en una red de flujo $\mathcal{G} = (V, E)$ con fuente s y sumidero t, entonces las siguientes condiciones son equivalentes:*

1. *f es un flujo máximo en $\mathcal{G}$.*
2. *La red residual $\mathcal{G}_f$ no contiene rutas de aumentación.*
3. *$|f| = c(S, T)$ para algún corte (S, T) de $\mathcal{G}$.*

Donde:

[3] La notación $|\cdot|$ indica que se trata del valor del flujo, no de un valor absoluto o de cardinalidad.

- *Red residual*. Es una red que consiste de aristas que pueden admitir más flujo de red. Formalmente, dada una red de flujo $\mathcal{G} = (V, E)$ y un flujo f, la red residual de $\mathcal{G}$ inducida por f es $\mathcal{G}_f = (V, E_f)$, donde

$$E_f = \{(u, v) \in V \times V : c_f(u, v) > 0\}$$

  y c_f es la capacidad residual de alguna ruta de aumentación, tal como se define en seguida.

- *Ruta de aumentación*. Dada un red de flujo $\mathcal{G} = (V, E)$ y un flujo f, una ruta de aumentación π es una ruta simple de s a t en la red residual $\mathcal{G}_f$. Esto significa que cada arista (u, v) sobre una ruta de aumentación admite algún flujo de red positivo de u a v sin violar la restricción de capacidad de la arista. La *capacidad residual* c_f de π es la máxima cantidad de flujo de red que se puede transportar a lo largo de las aristas de la ruta de aumentación π, y está dada por

$$c_f(\pi) = \min\{c_f(u, v) : \ (u, v) \ \text{está sobre} \ \pi\}$$

- *Corte de una red de flujo*. Un corte (S, T) de un red de flujo $\mathcal{G} = (V, E)$ es una división de V en S y $T = V - S$ de modo que $s \in S$ y $t \in T$. Si f es un flujo, entonces el *flujo de red* a través del corte (S, T) se define como $f(S, T)$. La *capacidad* del corte (S, T) es $c(S, T)$. Se puede demostrar que $f(S, T) = |f|$; es decir, el valor de un flujo es igual al del flujo de red hacia el sumidero. Más aún, el valor de cualquier flujo f en la red de flujo $\mathcal{G}$ está limitado hacia arriba por la capacidad de cualquier corte de G.

El Teorema 2.1 nos dice que un flujo es máximo si y sólo si su red residual no contiene rutas de aumentación. Que la máxima cantidad de flujo en una red es igual a la capacidad de su corte mínimo. El teorema *flujo máximo – corte mínimo* permite fundamentar los resultados del grafo de flujo de información descrito a continuación.

Grafo de flujo de información

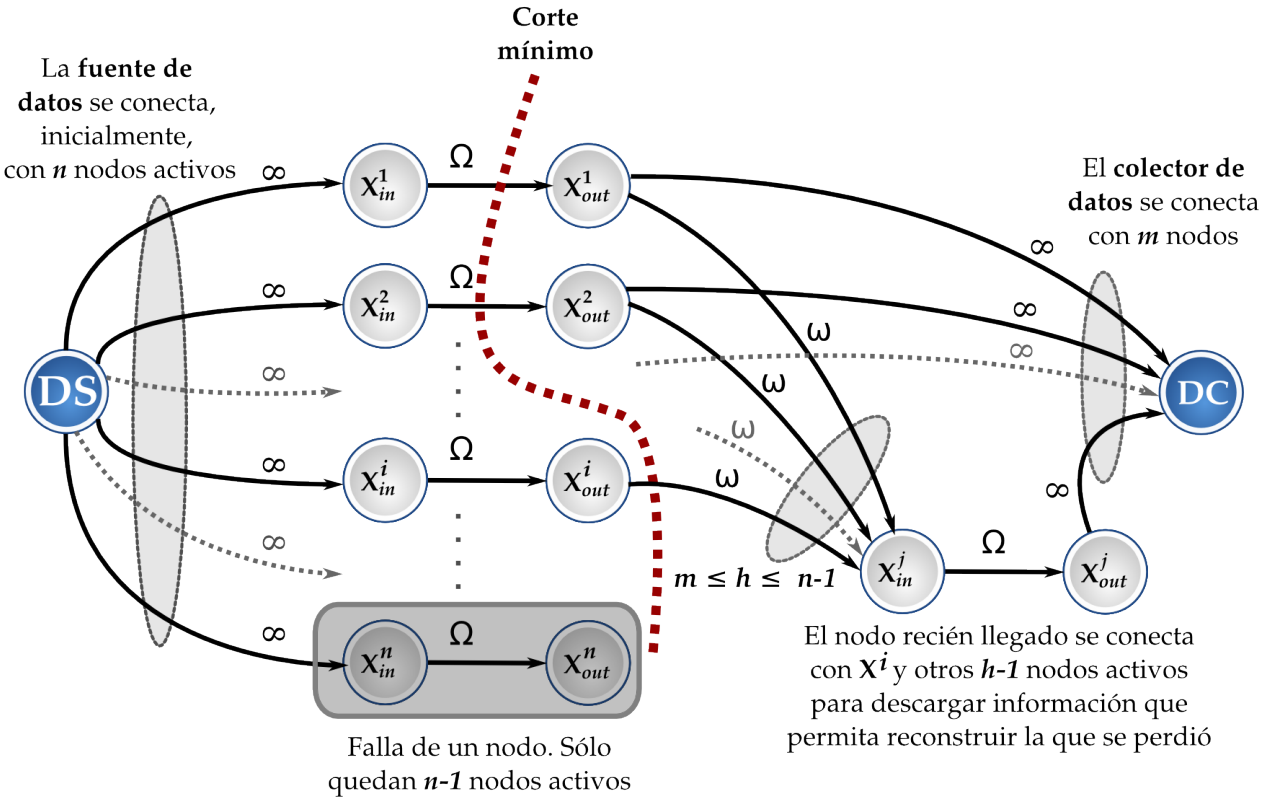
Para modelar los límites, en ancho de banda y almacenamiento, de un sistema de almacenamiento distribuido, en [23] se creó un *grafo de flujo de información* que cuenta son las siguientes características (Fig. 2.10):

- Es un grafo acíclico dirigido, denotado como $\mathcal{F}$, que representa un sistema de almacenamiento distribuido. En particular, describe cómo la información se almacena y se reconstruye en el tiempo.

- Cuenta con tres tipos de nodos:

  1. Un nodo DS productor de información original que se conecta a n nodos de almacenamiento mediante aristas de peso infinito. Esto es, $DS \xrightarrow{\infty} X^i_{in}$. Donde DS hace las veces del nodo fuente s descrito anteriormente.
  2. Un conjunto de n nodos de almacenamiento, cada uno representado por los pares X^i_{in}, X^i_{out}, con $1 \leq i \leq n$, conectados mediante aristas cuyo peso Ω es la cantidad de información almacenada en el nodo X^i. Es decir, $X^i_{in} \xrightarrow{\Omega} X^i_{out}$.
  3. Un colector de datos DC que representa una petición de reconstrucción de información, para lo cual debe conectarse con m nodos de almacenamiento. La arista que conecta al DC con los m nodos tiene un peso infinito. Esto es, $X^i_{out} \xrightarrow{\infty} DC$. Donde DC hace las veces del nodo sumidero t descrito anteriormente.

Figura 2.10: *Grafo de flujo de información $\mathcal{F}$ que modela la distribución de información en un sistema de almacenamiento distribuido.*



- Los nodos pueden encontrarse activos o inactivos. De modo que si un nodo j llega al sistema, éste sólo puede conectarse con los nodos activos. En caso de que el nodo j quiera conectarse con el nodo i, entonces se agrega una arista dirigida de X^i_{out} a X^j_{in} con un peso igual a la cantidad de datos ω que j recibe de i. Esto es, $X^i_{out} \xrightarrow{\omega} X^j_{in}$. Además, si un nodo deja el sistema, se considera inactivo y desaparecen sus aristas hacia otros nodos.

El recién definido grafo $\mathcal{F}$, por ser también una red de flujo, cumple con las restricciones impuestas por el corte mínimo y el flujo máximo. En vista de esto, podemos afirmar lo siguiente:

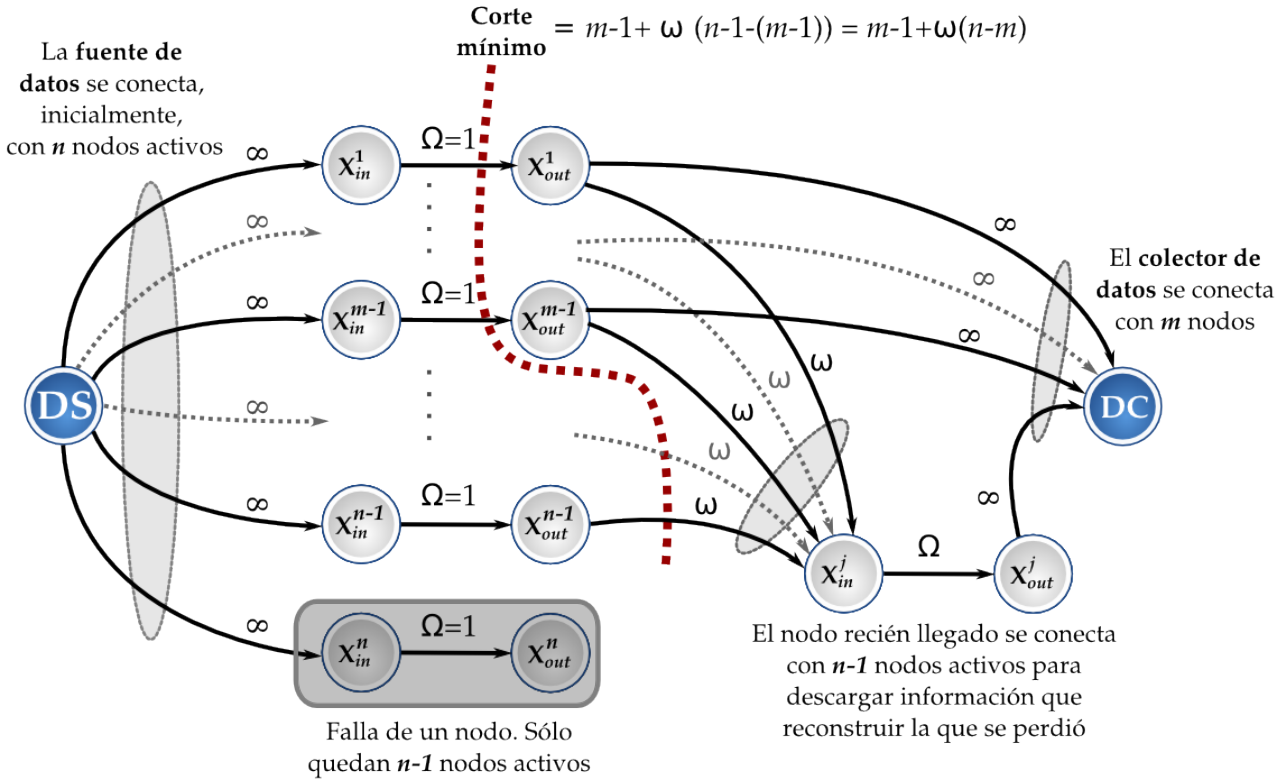
Lema 2.2 *Sea $\mathcal{F}$ el grafo de flujo de información de un sistema de almacenamiento distribuido con n nodos de almacenamiento. Entonces no se puede reconstruir la información generada en la fuente DS si la capacidad del corte mínimo $c(DS, DC)$ en $\mathcal{F}$ es menor que el tamaño inicial $|F| = M$ de la información original.*

Dicho en otras palabras, el Lema 2.2 indica que el flujo máximo en $\mathcal{F}$ —el que resulta de un corte mínimo— debe ser mayor o igual en tamaño que la información original para poder recuperarla. Esto se deriva del Teorema 2.1.

Proposición 2.3 *Sea $\mathcal{F}$ el grafo de flujo de información de un sistema de almacenamiento distribuido con n nodos de almacenamiento. Si elegimos la menor de las capacidades de los cortes mínimos obtenidos con todos los posibles $\binom{n}{m}$ colectores de datos DC, y se cumple el Lema 2.2, entonces todos los DC pueden recuperar la información original mediante algún código de red lineal —inclusive aleatorio, con alta probabilidad.*

En [23] se demuestra la Proposición 2.3 tomando como referencia los resultados de [3], donde se comprueba que los códigos de red pueden lograr el límite impuesto por el teorema *flujo máximo – corte mínino*.

Figura 2.11: *Grafo de Flujo de Información para un código de borrado tipo IDA (§ 2.3.2).*



Proposición 2.4 *Sea el archivo F distribuido por un sistema de almacenamiento distribuido mediante una codificación IDA (§ 2.3.2), es decir, un archivo de tamaño M dividido en m piezas, a partir de las cuales se genera un código de borrado (n, m) con n piezas codificadas de tamaño $\frac{M}{m}$; donde cada una de las n piezas codificadas se almacena en n diferentes nodos. Un nodo recién llegado X^j crea una nueva pieza codificada al descargar ω por ciento de cada $h = n - 1$ nodos de almacenamiento activos. Entonces*

$$\omega \geq 1/(n-m)$$

es necesaria y suficiente para reconstruir la información original y ofrece el mínimo ancho de banda posible para mantener un código de borrado del tipo MDS, tal como el IDA.

Demostración La Fig. 2.11 muestra el trazo del corte mínimo para el caso en el que el nodo recién llegado X^j se conecta a $h = n - 1$ nodos para descargar un porcentaje ω de información de cada uno de ellos. Cada nodo almacena $\Omega = 1$ unidades de información; de este modo, n nodos pueden almacenar las n partes que componen un archivo codificado. La capacidad del corte mínimo debe ser mayor o igual a la cantidad m piezas de información con la que se puede recuperar un archivo F distribuido con un código de borrado como el utilizado en § 2.3.2. Entonces, el corte mínimo en $\mathscr{F}$ es:

$$m - 1 + \omega(n - 1 - (m - 1)) \geq m$$

Donde el ω necesario para cumplir esa desigualdad es:

$$\omega \geq 1/(n-m)$$

■

Si el nodo recién llegado sólo se conecta a $h = m$ nodos de almacenamiento, el corte mínimo es $m - 1 - \omega$; con la restricción de que éste debe ser mayor o igual a m; de ser así entonces tenemos que $\omega = 1$. Lo anterior significa que el recién llegado tendría que descargar un fragmento íntegro de cada nodo para recuperar la información original.

Ahora bien, puesto que un código de borrado del tipo MDS no puede mantenerse con menor ancho de banda que en el caso de $h = n - 1$, éste sólo se podría mejorar con otro esquema de codificación. En [23] proponen el uso de códigos de red mediante una estrategia que ellos llaman *Regenerating Codes* (RC), los cuales tienen poco exceso en el uso de ancho de banda, aun cuando en la reconstrucción un nodo sólo se conecta a m nodos. Los RC pueden conseguir esta mejora porque permiten que los nodos recién llegados *almacenen todos los datos* que descargaron en vez de eliminarlos. De esta manera, si lo nodos mantienen información con ellos, entonces se pueden convertir en nodos que participan de un sistema de almacenamiento que utiliza códigos de red, donde la información es el resultado de las codificaciones que se hicieron en cada nodo incluido en cualquier ruta entre DS y DC.

Teorema 2.5 *Sea $\mathscr{F}$ el grafo de flujo de información de un sistema de almacenamiento distribuido con n nodos de almacenamiento. Si asumimos que todos los nodos almacenan ωM bits y los recién llegados se conectan a m nodos activos, de los que descargan $\frac{1}{m}\omega M$ bits de cada uno. Entonces, se define*

$$\omega_c = \frac{1}{m} \times \frac{1}{1 - \frac{1}{m} + \frac{1}{m^2}}$$

Demostración Si $\omega < \omega_c$, la capacidad del corte mínimo entre la fuente DS y algún subconjunto de m nodos de almacenamiento será menor que el tamaño M del archivo F y por lo tanto la reconstrucción no será posible. Si $\omega \geq \omega_c$, entonces la capacidad del corte mínimo será mayor o igual a M y, por la Proposición 2.4, existe un código de red lineal tal que todos los colectores de datos DC pueden recuperar la información original —aun aleatorio con alta probabilidad al incrementar arbitrariamente el tamaño del campo finito que lo defina (Fig. 2.12).

Por lo tanto es suficiente encontrar un ω_c tal que cualquier subconjunto m de nodos de almacenamiento tenga un corte mínimo con la fuente DS igual a M. Se procede mediante inducción de n, el número total de nodos de almacenamiento. Donde cualquier subgrafo de $\mathscr{F}$ con m entradas y $j \geq m$ salidas, denotado como $\mathscr{F}_j$, es un grafo *adecuado* si cada m de j salidas ofrece un flujo máximo de M. Para el caso base de la inducción asumimos que inicialmente hay m nodos de almacenamiento.

Por el paso inductivo, asumimos que tenemos un grafo adecuado $\mathscr{F}_{j-1}$ y un nodo recién llegado X^j se conecta a cualquiera m salidas de $\mathscr{F}_{j-1}$ con aristas con capacidad $\omega\frac{M}{m}$. Se debe demostrar que un nuevo grafo con las salidas de $\mathscr{F}_{j-1}$ más la salida X^j_{out} será el grafo adecuado $\mathscr{F}_j$.

Sea $C(X^j)$ el conjunto de nodos de almacenamiento a los que X^j está conectado. Consideramos que un colector de datos DC se conecta a un subconjunto de nodos c_1 de $C(X^j)^c$, a c_2 nodos de $C(X^j)$ y al recién llegado X^j. Todos los colectores de datos que no se conectan al recién llegado reciben suficiente flujo por la hipótesis de inducción. Por lo tanto tenemos que $c_1 + c_2 = m - 1$ y la capacidad del corte mínimo para este colector de datos es:

$$c_1\omega M + c_2\omega M + (m - c_2)\frac{\omega M}{m},$$

que, bajo la restricción de que debe ser mayor o igual a M:

$$c_1\omega M + c_2\omega M + (m - c_2)\frac{\omega M}{m} \geq M,$$

para todo

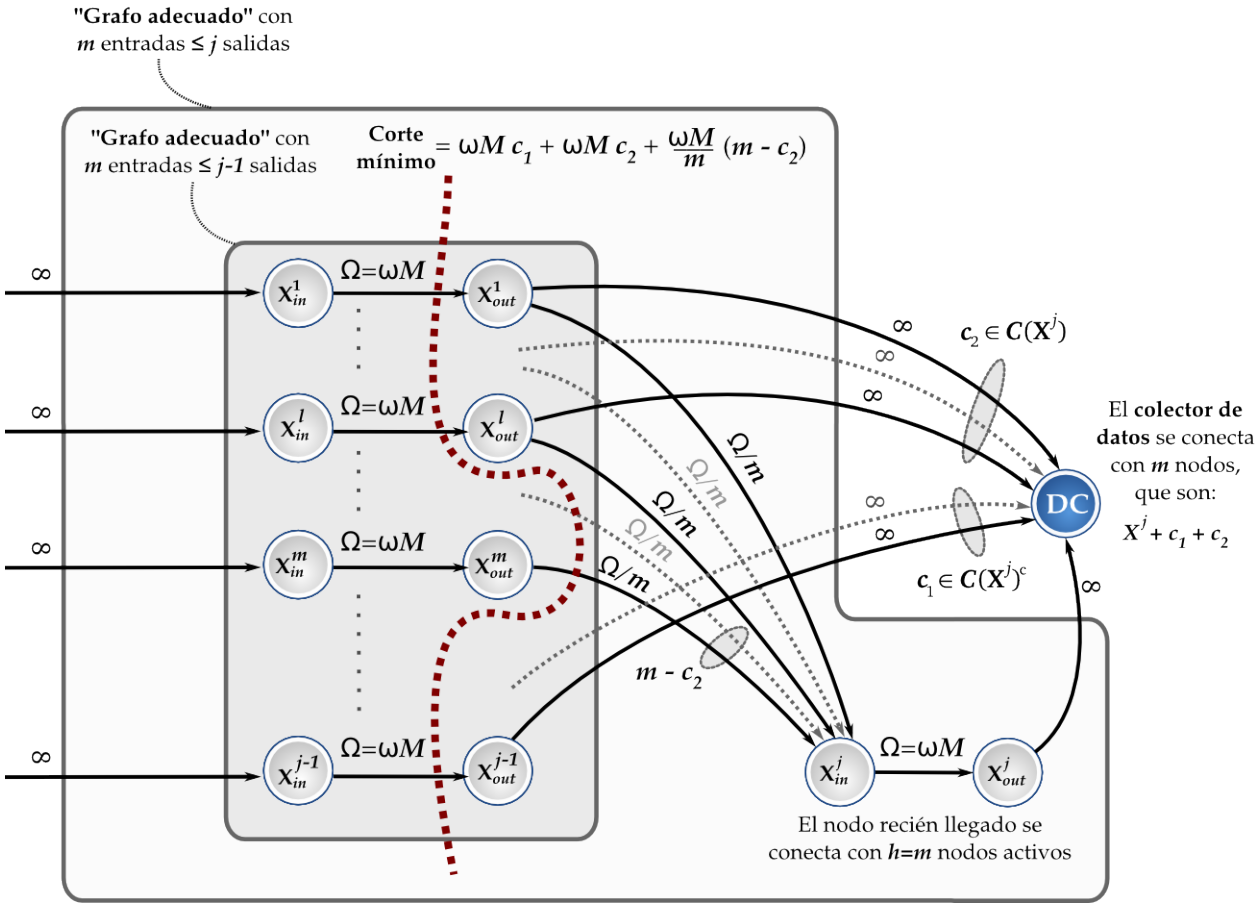
$$c_1, c_2, c_1 + c_2 = m - 1.$$

Sabemos que el *peor caso* —en el que se utiliza más ancho de banda, cuando $h = m$—, se da con $c_1 = 0$, de esto se obtiene que

$$\omega \geq \frac{1}{m} \times \frac{1}{1 - \frac{1}{m} + \frac{1}{m^2}} = \omega_c$$

■

Figura 2.12: *Grafo de Flujo de Información para códigos de red mediante* Regenerating Codes.



Capítulo 3

Sistemas de Almacenamiento P2P

Antes de discutir una definición y justificación de los sistemas de almacenamiento P2P, a continuación ofrecemos una introducción a los sistemas P2P que también puede servir de fundamento y referencia para las secciones restantes.

3.1. Sistemas P2P

Los sistemas P2P no son un concepto nuevo en la historia de las telecomunicaciones. Durante una llamada telefónica, por ejemplo, dos personas comparten estatus y una conexión punto a punto, lo mismo que en un sistema P2P. Internet, por su parte, inició como un sistema P2P [89] donde los primeros equipos en la periferia (*hosts*) eran parte de una misma jerarquía dentro de la red ARPANET. Asimismo, la mayoría de las primeras aplicaciones distribuidas pueden considerarse como P2P, tal como los sistemas de correo-e basados en el protocolo SMTP, o el conjunto de programas UUCP para la ejecución remota de comandos, transferencia de archivos y contenido —e.g., noticias—, que evolucionaron en las importantes redes Usenet y FidoNet.

Con la llegada del módem y las computadoras personales sobrevino un extraordinario aumento en el uso de Internet como un medio de comunicación, en primera instancia, y de distribución de datos, como consecuencia. Es en estas condiciones que en 1999 aparece Napster [62, 54], la aplicación tomada como referencia en prácticamente todos los estudios sobre P2P, y la que abrió nuevas posibilidades en el uso —y abuso— de Internet con la distribución masiva de archivos MP3 entre sus usuarios.

A la fecha, los sistemas P2P han evolucionado en la calidad y cantidad de sus aplicaciones, muchas de ellas ya traspasan el ámbito del usuario común y llegan a solucionar complejas problemáticas de índole científico y empresarial. En lo que sigue presentamos la características más importantes de estos sistemas, las definiciones que los acompañan y dos clasificaciones esenciales para comenzar a abordar un tema tan amplio.

3.1.1. Definición

En esta sección señalaremos las que consideramos como las características más importantes de los sistemas P2P y algunas de las definiciones que los engloban. El objetivo principal de los sistemas P2P es *compartir* recursos de cómputo —e.g., ciclos de CPU, almacenamiento secundario, ancho de banda de la red de cómputo, y contenidos—, de manera distribuida. Sin embargo, esa sola capacidad para compartir no los diferencia de otros sistemas de cómputo como los sistemas de archivos distribuidos —e.g., NFS, CIFS, CODA— o los *middleware* para hacer cómputo paralelo —e.g., MPI, Linda—; es por eso que a continuación exponemos las características más distintivas e importantes de los sistemas P2P:

- *Arquitectura de procesos.* No existe la tradicional arquitectura cliente/servidor, donde uno de los participantes del sistema —el *cliente*— hace solicitudes de servicio a otro participante —el *servidor*—, quien las atiende, procesa y entrega en correspondencia a esos estímulos. En los sistemas P2P, en cambio, cada participante es un *igual*, un *par*, con capacidades y jerarquía idénticas para solicitar y proveer servicios de cómputo: cliente y servidor a la vez, un *servent* —*server* y *client*—, según se propone en la jerga del sistema P2P Gnutella. Normalmente, cada proceso está asociado únicamente a un *host* dentro del sistema P2P, esto nos lleva a deshacer, para fines expositivos, la distinción entre *host* y proceso P2P. De hecho, de aquí en adelante nombraremos simplemente *nodo* al equipo de cómputo o proceso que es miembro de algún sistema P2P.

- *Autonomía.* No exite un control centralizado, i.e., una entidad de cómputo con mayor jerarquía dentro del sistema que administre las actividades de las demás. Cada nodo es autónomo y depende de sí mismo para regular su ciclo en vida en el sistema: ingresar, participar y abandonar. En contraparte, cada nodo autónomo depende de otros nodos autónomos para obtener información, recursos de cómputo, reexpedir peticiones, etc.

- *Autoorganización.* Los participantes del sistema P2P, en tanto que son autónomos, también crean las condiciones necesarias para organizarse sin el control de un agente externo y formar así un sistema de cómputo distribuido con la capacidad de autosustentarse y cumplir con los objetivos que le dan razón de ser. En este sentido, también se dice que los sistemas P2P son sistemas orgánicos.

- *Escala.* Los sistemas P2P son sistemas de cómputo masivamente distribuidos que pueden y deben crecer de manera natural, gracias a su capacidad de autoorganización, a escalas que no tienen precedentes en otros sistemas distribuidos.

- *Arquitectura de red.* Para obtener características como autonomía y autoorganización, los sistemas P2P crean una red lógica, llamada *red superpuesta* (§ 3.1.2), sobre la red física existente que les permita colaborar y comunicarse conforme a sus necesidades y propósitos.

Las características descritas arriba son generales y de ellas se derivan otras de gran importancia: alta disponibilidad de los recursos, tolerancia a fallas, reducción de costos en función del grado de distribución, agregación de recursos a gran escala, interoperabilidad, dinamicidad, comunicación y colaboración *ad hoc* entre usuarios en los extremos de Internet, etcétera. Fundadas en ellas, podemos hallar una cantidad grande y diversa de definiciones de los sistemas P2P en la literatura; estas son algunas:

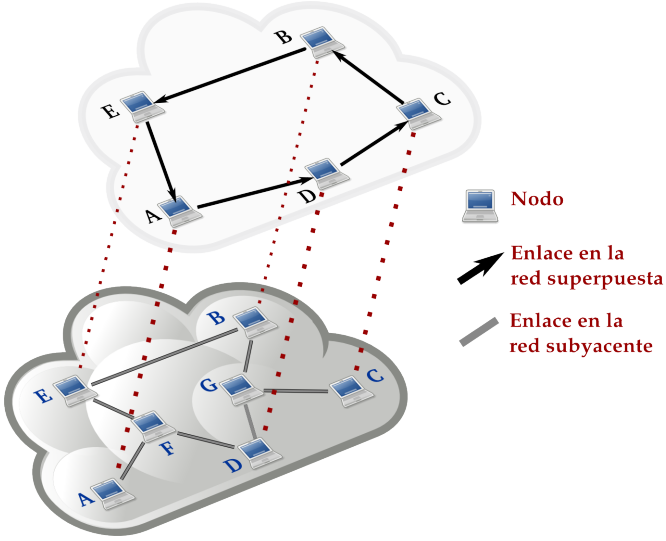
> «Los sistemas P2P son sistemas distribuidos que consisten de nodos interconectados capaces de autoorganizarse en topologías de red con el propósito de compartir recursos tales como contenido, ciclos de CPU, almacenamiento y ancho de banda, capaces de adaptarse a las fallas y acomodar poblaciones temporales de nodos mientras mantienen niveles de conectividad y desempeño aceptables, sin requerir de intermediación o apoyo de una autoridad o servidor centralizado [7].»

> «P2P es una manera de estructurar aplicaciones distribuidas de modo que los nodos individuales que componen el sistema tengan roles simétricos [64]».

> «Las redes P2P son aquellas que exhiben tres características: autoorganización, comunicación simétrica y control distribuido [78].»

> «P2P es una clase de aplicaciones que toman ventaja de los recursos —de almacenamiento, ciclos de CPU, contenido, presencia humana— disponible en las fronteras de Internet [63].»

Figura 3.1: *Red superpuesta* [30].



«Los sistemas P2P son sistemas masivamente distribuidos y altamente volátiles para compartir grandes cantidades de recursos [13].»

«P2P se refiere a una clase de sistemas y aplicaciones que emplean recursos distribuidos para realizar una función crítica de manera descentralizada. Los recursos comprenden poder de cómputo, datos —almacenamiento y contenido—, ancho de banda de la red, y presencia —computadoras, humanos, y otros. La función crítica puede ser cómputo distribuido, distribución de datos y contenido, comunicación y colaboración, o servicios de la plataforma. La descentralización puede aplicarse a algoritmos, datos y metadatos, o bien, a todos ellos [55].»

Aquí es justo mencionar que consideramos que definición propuesta por [7] es la más completa en términos de las características descritas con anterioridad y la que tomaremos como referencia en este trabajo de investigación.

3.1.2. Redes superpuestas

Las redes superpuestas crean una topología virtual sobre los protocolos básicos de transporte [25]; también es el término utilizado por [11] para describir la configuración dentro de la cual una red de base es usada para servir de soporte a una segunda red. Una capa arriba de otra, de ahí el término *superpuesta*; en el mismo sentido, la red básica suele nombrarse *red subyacente*. La Fig. 3.1 muestra una red superpuesta sobre una red subyacente, donde cada una tiene el potencial de mostrar toda una serie de topologías y funcionalidades distintas entre sí.

Internet comenzó como una red superpuesta encima de la red telefónica, usando enlaces telefónicos para conectar encaminadores de paquetes. Igualmente, una red superpuesta se vale de las máquinas en los extremos de Internet para construir los enlaces y encaminadores que conforman una nueva red que no requiere la actualización de la existente para agregar nuevas funcionalidades. El proyecto PlanetLab[1], por ejemplo, provee una infraestructura donde se llevan a cabo una amplia gama de trabajos experimentales con redes superpuestas.

[1] http://www.planet-lab.org.

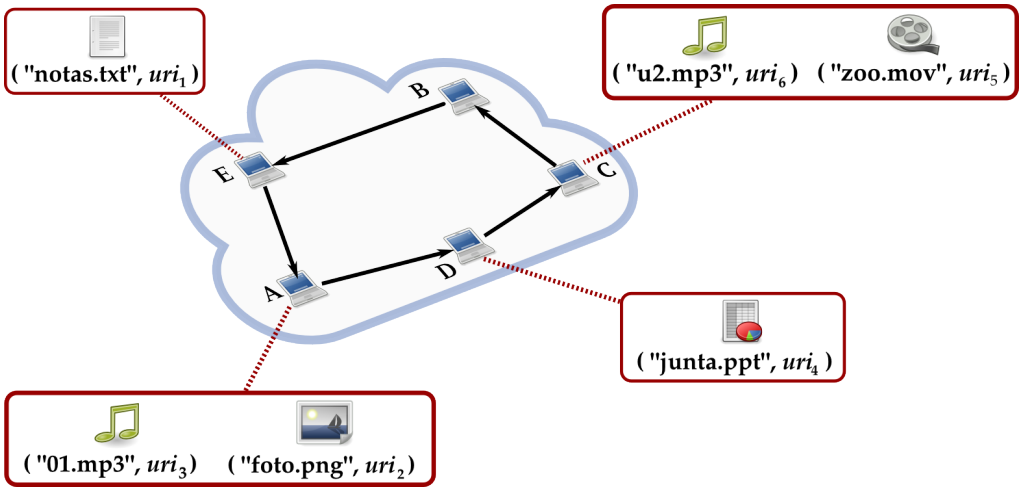
Existen dos tipos de redes superpuestas [5]: de *encaminamiento*, para mejorar o reemplazar el provisto por la red subyacente, y de *almacenamiento y búsqueda*, para aprovechar la capacidad de grandes colecciones distribuidas de equipos de cómputo. En general, esos dos tipos de redes se mezclan para lograr redes superpuestas más especializadas y robustas.

Cada red superpuesta asocia un identificador global único a sus elementos; algún punto de acceso para que más elementos puedan agregarse, y un conjunto de propiedades garantizadas [11] —simples, en comparación con las ofrecidas por la red básica—. En resumen, podemos distinguir cuatro propiedades fundamentales de las redes superpuestas para construir redes P2P:

1. Obtención de datos garantizada.
2. Tiempos de búsqueda acotados (típicamente $O(\log \mathcal{N})$, donde $\mathcal{N}$ es el número de nodos de la red).
3. Balance de carga automático.
4. Autoorganización.

Una red superpuesta que ha merecido una buena cantidad de trabajos de investigación es la llamada *tabla hash distribuida*, o DHT, por sus siglas en inglés. Ésta es una tabla *hash* repartida entre un conjunto de computadoras cooperantes [30], que construye una red superpuesta estructurada para encaminamiento y búsqueda. Al igual que una tabla *hash*, contiene pares *llave*/*valor* para ofrecer el servicio de búsqueda de datos a través de una interfaz con las operaciones `obtener`(*llave*) y `establecer`(*llave*, *valor*).

Figura 3.2: *Mapeo de nombres de archivo a URIs en una DHT* [30].



En el ejemplo de la Fig. 3.2, la DHT mapea nombres de archivos a sus ubicaciones actuales, físicas, representadas por los URI [2]. Los elementos de la DHT están distribuidos entre los nodos A, B, C, D y E; con apuntadores de ruta para encaminar peticiones entre sí. Si una aplicación hace una petición por el archivo "`junta.ppt`" desde E, este nodo encaminará la petición al nodo A, el cual hará lo mismo hacia D, quien constestará favorablemente pues conoce la URI asociada a "`junta.ppt`". No todos los nodos almacenan información, e.g., el nodo B [30]. Una DHT implementa los siguientes componentes básicos:

1. *Mapeo/direccionamiento entre datos y nodos*. En otras palabras, la aplicación de una función de *hash* para soportar búsquedas, que puede ir desde una URI mapeada a una dirección IP, hasta la opción

[2] Localizador Uniforme de Recursos.

Tabla 3.1: *Tipos de DHT*

Tipo	Implementación	Año
Árbol de Plaxton	Tapestry [93]	2001
Hipercubo	CAN [71]	2001
Anillo	Chord [86]	2001
Mariposa	Viceroy [50]	2002
XOR	Kademlia [53]	2002

más aceptada: aplicar la función SHA-1 para obtener una llave que mapee la dirección de los datos a un nodo particular, esta técnica soporta uniformidad y seguridad. Además, se aplica *hash* consistente (ver más abajo) para distribuir equitativamente datos entre los nodos.

2. *Protocolo de encaminamiento*. Ejecutado sobre la correspondiente red superpuesta, permite la búsqueda de datos: el proceso más importante de una DHT. Los protocolos de encaminamiento deben proveer búsquedas eficientes y al mismo tiempo minimizar la necesidad de conocer el estado de los nodos.

3. *Protocolo de mantenimiento del estado de las rutas*. Cada nodo en la DHT posee un estado que es utilizado para mantener la tablas de encaminamiento de los nodos actualizadas y consistentes, a pesar de la dinamicidad del sistema. Éste es el gran reto de una DHT.

La Tabla 3.1 muestra diferentes diseños de DHT, primera implementación y año de publicación. Después de esos diseños básicos, se han propuesto otros más complejos y especializados a partir de ellos como Pastry [79] —una combinación de Chord y Tapestry—, DHash++ [21], One-Hop [38], Kelips [47], D1HT [56], etcétera. Las primeras DHT comenzaron a aparecer en el 2001, y se basaron en alguna de las dos ideas siguientes:

1. *Hash* consistente [42]: Utilizado para mapear de manera *uniforme* llaves a valores, permitiendo así un balance de carga natural al asignar a cada nodo prácticamente el mismo número de llaves.

2. *Plaxton mesh* [69]: Permite el encaminamiento eficiente al nodo responsable de un objeto dado, al tiempo que requiere un tabla de encaminamiento relativamente pequeña.

Del 2001 a la fecha, se ha realizado una gran cantidad de trabajos de investigación encaminados a diseñar y producir DHT cada vez más eficientes y flexibles. Ya no sólo experimentos bajo escenarios controlados, sino también verdaderos sistemas que mediante una API permitan la construcción de aplicaciones P2P con relativa facilidad, con bajo costo de desarrollo y mantenimiento. Quizás el esfuerzo más avanzado en ese sentido es OpenDHT [72].

3.1.3. Clasificación

Por jerarquía de procesos

Iniciamos con la Fig. 3.3, donde se muestra la ubicación de los sistemas P2P en relación a otros sistemas de cómputo. Claramente, los sistemas P2P son sistemas distribuidos que a su vez pueden considerarse *puros* o *híbridos*, en función de existencia o no de jerarquías entre los nodos, respectivamente. En este sentido, Napster es un sistema híbrido, mientras que Gnutella [34] es un sistema P2P puro por excelencia.

Napster mantenía un servidor con un directorio centralizado al que los nodos enviaban su dirección y la lista de archivos MP3 que compartirían. Los nodos sometían búsquedas —e.g., nombre de canciones,

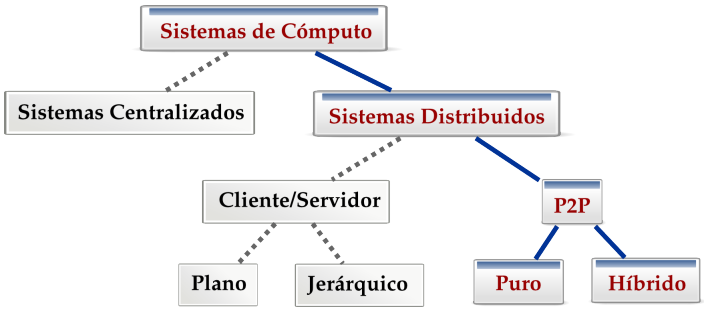


Figura 3.3: *Taxonomía de los sistemas de cómputo* [55].

intérpretes— a ese directorio y éste les devolvía una lista con direcciones IP. El directorio centralizado evitaba problemas de búsqueda que son comunes en otros sistemas P2P pero fácilmente se convertía en un «cuello de botella» y en único punto de falla. A partir de Napster se desprendieron otros P2P híbridos como Groove, Aimster, Magi, Softwax, iMesh, etc.

Gnutella, opuesto arquitectónico de Napster, es un sistema totalmente descentralizado donde los nodos nuevos se conectan a uno conocido de antemano y luego se dan a conocer a sus vecinos mediante una técnica de inundación por mensajes (*flooding*). Se dice que la red Gnutella es un sistema P2P puro porque, en efecto, todos los participantes en ella tienen igual jerarquía y funciones.

Las redes Gnutella, aunque numerosas, nunca alcanzaron el éxito de las híbridas mencionadas arriba, quizá por el uso ineficiente del ancho de banda de la red, pero sobre todo porque la topología de la red formada por los nodos es tan irregular, poco estructurada, que es muy difícil garantizar el éxito de una búsqueda; de un archivo MP3, por ejemplo.

Con el tiempo Napster desapareció en medio de pleitos legales y Gnutella se convirtió en el protocolo utilizado por muchas aplicaciones de código abierto, esto lo ha llevado a convertirse en objeto frecuente de estudio [73, 17, 74] .

Kazaa [43], por su parte, utilizó un enfoque intermedio: súpernodos con información que otros no tienen. Estos súpernodos son análogos al servidor de índices utilizado por Napster, sólo que a menor escala. Kazaa llegó a rebasar la popularidad de Napster, aunque actualmente es más conocido por ser un distribuidor masivo de virus de computadoras y software espía.

Por topología de la red

En las siguientes subsecciones mostraremos una clasificación de los sistemas de almacenamiento P2P y luego ofreceremos una visión general de los temas más importantes al respecto, con particular atención en las estrategias de redundancia de información. Como todo sistema P2P, los de almacenamiento pueden clasificarse según la topología que forman los pares [13].

- **No estructurados**

  Estos se refieren a los sistemas P2P que no tienen restricciones sobre dónde ubicarse en la red. Estos sistemas se caracterizan por utilizarse para la distribución masiva de archivos. No necesariamente son sistemas P2P puros —e.g., Gnutella, Freenet—, sino que pueden formarse jerarquías —e.g., supernodos FastTrack y Kazaa, servidor Napster—, o ser sistemas intermedios: puros, con alguna débil jerarquía —e.g., BitTorrent—. La duplicación de archivos populares aumenta su disponibilidad y la suma de ancho de banda para descargarlos. Las redes P2P no estructuradas forman topologías prácti-

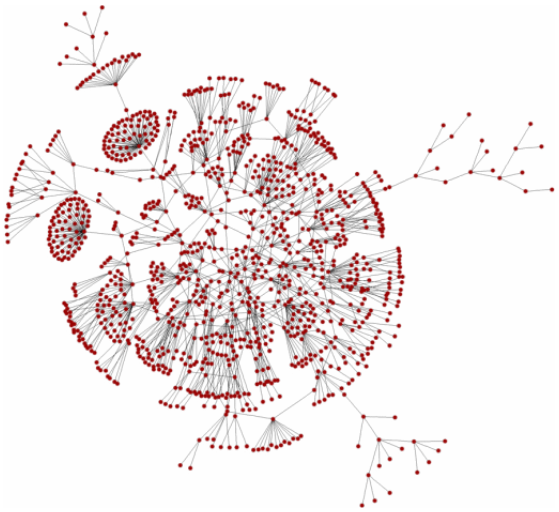


Figura 3.4: *Topología de una red Gnutella generada con GnuMap* [15].

camente aleatorias, fáciles de mantener, pero donde el costo de las búsquedas no puede garantizarse (Fig. 3.4). En la red Gnutella, por ejemplo, es difícil garantizar que un archivo sea encontrado.

- **Estructurados**

  Estos sistemas P2P proveen garantías y límites en los tiempos de búsqueda y almacenamiento, con el compromiso de mantener una rígida estructura de red. En realidad se trata de redes superpuestas (§ 3.1.2) construidas a partir de tablas *hash* distribuidas (§ 3.1.2), por lo que la topología de la red corresponde a la DHT utilizada (Tabla 3.1 y Fig. 3.5). En su forma más básica, los sistemas P2P estructurados son sistemas de localización de datos a nivel de la capa de aplicación que pueden convertirse en la infraestructura de un sistema de búsqueda más complejo.

La Tabla 3.2 muestra el compromiso entre el costo de mantenimiento vs. costo de búsqueda en las redes P2P de los dos tipos mencionados arriba. La dinamicidad de los sistemas P2P complica aún más dicho compromiso debido a las dificultades que existen para su mantenimiento.

Tabla 3.2: *Compromiso entre redes P2P estructuradas y no estructuradas*

Tipo	Costo de Mantenimiento	Costo de búsqueda
Estructurado	Alto	Bajo (determinístico)
No estructurado	Bajo	Alto (pocas garantías)

Por tipo de aplicaciones

Las aplicaciones basadas en sistemas P2P [3] pueden clasificarse en cuatro grandes grupos: cómputo distribuido, administración de archivos y contenido, colaboración y plataformas de desarrollo (Fig. 3.6). En los siguientes párrafos hacemos una breve descripción de cada uno de ellos.

[3] O'Reilly mantiene una extensa lista de aplicaciones que usan tecnologías P2P que, aunque obsoleta, refleja la variedad de aplicaciones de dichos sistemas: *http://www.openp2p.com/pub/q/p2p_category* (enero de 2009).

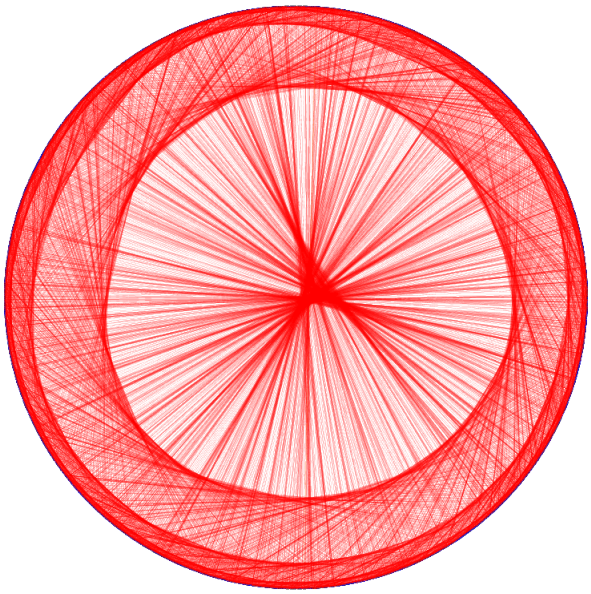


Figura 3.5: *Topología de una red Chord de* 1000 *nodos generada con PlanetSim* [68].

- **Cómputo distribuido**

  La agregación y distribución masiva de recursos de cómputo en los sistemas P2P dio paso a la creación de aplicaciones que explotaran este hecho. Por otro lado, el cómputo científico es una de las disciplinas más ávidas de ciclos de CPU para solucionar sus problemáticas, principalmente: simulaciones numéricas, análisis de datos y optimización de modelos matemáticos. El proyecto SETI@home [6, 80] ofreció en 1999 la primera intersección entre los sistemas P2P y el cómputo científico con el fin de detectar vida extraterrestre.

  Los participantes del proyecto SETI@home descargan el software desde el sitio del proyecto, el cual instalan y ejecutan como un salvapantallas o como cualquier otra aplicación de escritorio. La aplicación se conecta a un servidor central para descargar un conjunto de datos obtenidos por un radiotelescopio, mismos que trata con algoritmos de procesamiento de señales, para después enviar los resultados de vuelta al servidor central.

  Debido al éxito del proyecto, nacieron otros proyectos similares como Einstein@home, procesamiento de datos astronómicos; Rosetta@home, procesamiento de datos biológicos; LHC@home, simulación de partículas subatómicas aceleradas sobre el *Large Hadron Collider*; entre muchos otros, todos ahora construidos sobre una plataforma para cómputo distribuido llamada BOINC (*Berkeley Open Infraestructure for Network Computing*).

  Por su parte, otras áreas de estudio trabajan hacia una sinergia [88, 39, 26] entre los sistemas P2P y los *Grids* [27], a través de estándares como el OGSA (*Open Grid Services Architecture*) [28] y la plataforma Globus [33].

- **Administración de archivos y contenido**

  Un sistema de administración de archivos y contenido P2P crea un medio de almacenamiento distribuido que permite publicar, buscar y obtener archivos entre los nodos del sistema [7]. Las aplicaciones P2P para distribuir contenido —e.g., audio, vídeo, libros— son las más populares de todas.

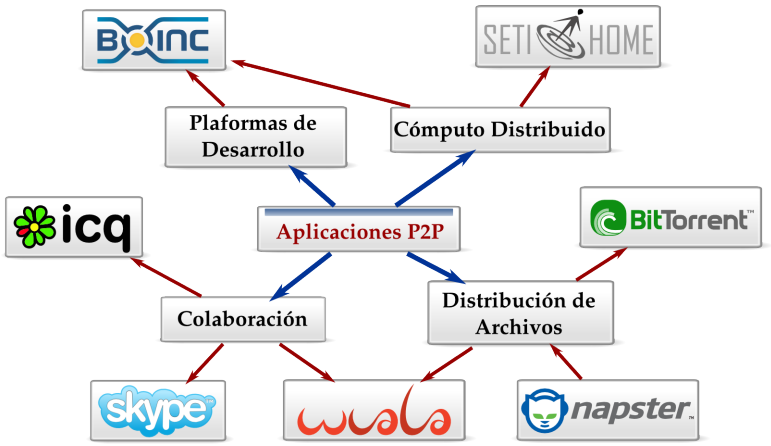


Figura 3.6: *Taxonomía de las aplicaciones P2P* [55].

Las aplicaciones para compartir sistemas de archivos, i.e., espacio en disco duro, son cada día más importantes en términos de investigación, desarrollo y negocios.

Como ya mencionamos arriba, Napster, Gnutella y Kazaa son aplicaciones/protocolos P2P para distribuir contenidos, pero existen otros como Bittorrent [19], que es un protocolo P2P en el que se basan muchas aplicaciones cliente —Azureus, μTorrent, BitComet, BitTornado— para la distribución masiva y eficiente de archivos.

Por otro lado, Freenet, Scan, Publius, Groove, Mnemosyne, FreeHaven, MojoNation, entre muchos otros, son o llegaron a ser, proyectos importantes en el desarrollo de tecnologías de distribución y administración de espacio de almacenamiento. Cada uno con enfoques distintos y especializados, como seguridad, anonimato, publicación, persistencia.

- **Colaboración**

  Las aplicaciones colaborativas son las que permiten a sus usuarios contribuir, en tiempo real, en la recopilación y retransmisión de información sin el auspicio de un servidor central [55]. Las aplicaciones de mensajería instantánea, edición de documentos en línea, juegos de vídeo en línea, entre muchas otras, son ejemplos de aplicaciones colaborativas.

  Una de las aplicaciones pioneras en los sistemas P2P es ICQ, una aplicación de mensajería instantánea que permite la comunicación directa entre pares a través de una conexión administrada por ellos, donde no existe una relación entre la dirección utilizada por ICQ y la dirección IP del servidor de nombres (DNS), pues crea una red lógica encima de la física. No obstante, como Napster, ICQ no es un sistema P2P puro porque aún existe un servidor que interviene en las conexiones entre pares antes de que éstas se establezcan. Otras aplicaciones P2P para mensajería instantánea más recientes son X-Com y P2P Messenger.

3.1.4. Herramientas de evaluación

Normalmente, los sistemas de cómputo deben ser probados y evaluados de alguna manera para caracterizarlos objetivamente y validar hipótesis sobre su funcionamiento. Esto incluye ofrecer a la comunidad resultados reproducibles para que los pueda comprobar. Históricamente, para esos propósitos, la ciencia se

ha valido de la teoría y la experimentación, pero la llegada de las computadoras a mitad del siglo XX agregó una nueva opción: la *simulación*. El estudio de los sistemas P2P no escapa a esas opciones, sin embargo, las características de estos sistemas impiden la viabilidad de algunas de ellas.

El enfoque teórico nos puede ofrecer el modelo matemático de algún sistema P2P, este mismo puede ser discreto o continuo, sin embargo su complejidad puede llegar a ser tal que suele resultar más conveniente hacer simplificaciones —en ocasiones excesivas—. Con la experimentación de sistemas P2P los problemas son aun mayores. Sabemos que algunas aplicaciones P2P pueden escalar a varios cientos de miles de usuarios, incluso millones. Si intentamos hacer un experimento de escala relativamente menor, con algunos cientos de máquinas, no sólo es costoso en términos económicos, sino también en los temporales.

En vista de los inconvenientes de los enfoques téorico y experimental, la simulación resulta un camino viable para evaluar y probar sistemas P2P a gran escala. De ahí que los simuladores P2P crezcan en su valor como herramientas de investigación. Sin embargo, como se argumenta en [60, 61], los simuladores P2P actuales tienen características indeseables, esto es:

- Proveen una funcionalidad incompleta que no cubre un amplio rango de posibilidades de experimentación.
- Ofrecen pobre documentación, la cual es indispensable para crear redes P2P complejas.
- Carecen de mecanismos robustos, flexibles, reproducibles, para reunir estadísticas.
- Ofrecen poca escalabilidad en lo referente a la cantidad de nodos —que refleja el tamaño del sistema— que pueden simularse.
- Ofecen pobre *usabilidad* y requieren largas curvas de aprendizaje.

Los simuladores más utilizados en la investigación de sistemas P2P generalmente son hechos a la medida [61], una situación que sólo ayuda a reafirmar las carencias descritas anteriormente. Aunque los simuladores NS-2, Javasim, P2PSim, PeerSim, Neurogrid tienen la capacidad de simular sistemas P2P, cada uno en diferentes niveles de abstracción, no son recomendados por [61]; recientemente, sin embargo, Oversim [8] parece remontar algunas carencias de sus predecesores.

Otro camino, que no requiere simuladores, es el uso de modelos computacionales para representar sistemas P2P a gran escala. Por ejemplo, pueden encontrarse en la literatura el uso de simulaciones con el método Monte-Carlo [4]. También existen las simulaciones basadas en *trazas*: estadísticas que resultan de rastrear las funcionalidades de algún sistema ejecutado en situaciones reales. Las trazas pueden inyectarse a algún modelo de un sistema P2P para ofrecerle condiciones de prueba similares a la experimentación. No obstante, conseguir trazas puede resultar una tarea difícil, particularmente de protocolos P2P con especificaciones cerradas como Kazaa y Skype, por mencionar algunos.

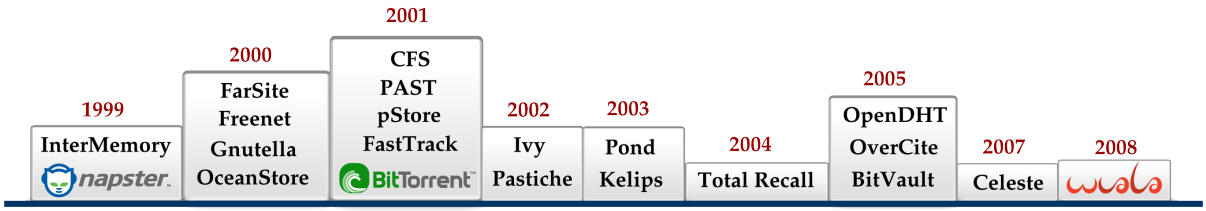
Otro punto de partida para la evaluación de sistemas P2P es construir uno original que cumpla con nuestros propósitos. Para esto existen plataformas de desarrollo que, en general, proveen componentes de software para programar aplicaciones P2P rápidamente y libres de errores [85]. Cada alternativa tiene su propio enfoque y diferentes niveles de apoyo al programador. Denominamos a esas plataformas de desarrollo como *middleware*, definido éste a su vez como la capa de software que yace entre el sistema operativo y la aplicaciones entre cada sitio del sistema [45]. JXTA, BOINC, Speakeasy, AntHill, son algunos de los *middlewares* más importantes.

3.2. Justificación y definición

La cada vez mayor capacidad de los dispositivos de almacenamiento [1, 65, 66], la consecuente cantidad de datos e información generada, aunadas al uso crecientemente masivo de Internet y el nacimiento de los

sistemas P2P, representaron la oportunidad para una siguiente generación de sistemas de almacenamiento: aquellos que permiten la distribución, persistencia, publicación y/o búsqueda de información entre los elementos —cada uno independiente de los demás— que lo conforman. A estos sistemas de cómputo les llamamos *sistemas de almacenamiento P2P*, de los cuales en la literatura podemos encontrar una buena variedad de propuestas de diseño. En la Fig. 3.7 se esboza una línea de tiempo de los sistemas de almacenamiento P2P.

Figura 3.7: *Línea del tiempo de los sistemas de almacenamiento P2P.*



La principal diferencia entre los sistemas de almacenamiento P2P y otros sistemas de almacenamiento distribuido, radica en el comportamiento de los nodos que integran la red P2P (§ 3.1.1). Normalmente, los sistemas de almacenamiento distribuido cuentan con una infraestructura de cómputo expresamente dispuesta para sus fines; mientras que la infraestructura P2P, como ya hemos mencionado, dista mucho de ser estable, dedicada o predecible. En general, el diseño de un sistema de almacenamiento P2P contempla los siguientes asuntos [40]:

- La *simetría* de roles y responsabilidades entre los nodos miembros del sistema. En el caso ideal, un sistema P2P puro, un nodo puede comportarse como servidor que almacena información y recibe peticiones; cliente que hace peticiones para almacenar y/o recuperar información, y encaminador para reexpedir mensajes a otros nodos en el sistema.
- La *participación natural* de los nodos, i.e., ésta no debe forzarse ni esperarse.
- La *robustez* ante la entrada y salida de nodos en el sistema. Cuando este comportamiento es intenso y oscilante, se le conoce como *churn*.
- La *rápida ubicación de recursos*, adaptable a una topología cambiante.
- El *balance de carga* para la óptima distribución de recursos.
- El *anonimato* para resistir censura.
- La *seguridad* para proteger datos ante ataques y fallas.
- La *disponibilidad* de la información (§ 2.1.2).

Los sistemas de almacenamiento P2P son también sistemas de descubrimiento y obtención de información. Sobre un sistema de almacenamiento debe existir un sistema de búsqueda cuya operación sea flexible y eficiente, con garantías en tiempo y calidad en los resultados, a pesar de la heterogeneidad que subyace en los elementos del sistema y su dinamicidad.

3.3. Etapas

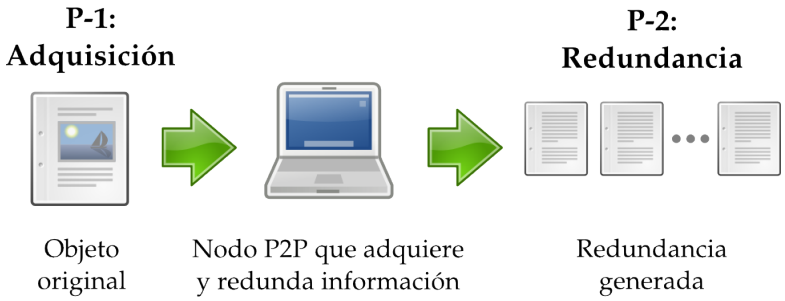
En este trabajo consideramos que las etapas que conforman el proceso de almacenar información en un sistema de almacenamiento P2P son tres: *Procesamiento*, *Distribución* y *Mantenimiento*. En cada una de ellas se asume la existencia de un sistema P2P ideal que, por diseño, incluye técnicas de búsqueda y análisis que le permiten realizar tareas importantes como la elección de nodos vecinos y/o activos, creación de la topología del sistema, ubicación de objetos de información y su redundancia, etcétera. También se asume que el espacio en disco y ciclos de CPU de cada nodo, así como el ancho de banda de la red de comunicaciones son recursos de los que se dispone en abundancia, pero cuya utilización debe cuidarse. Asimismo, representamos la información —e.g., archivos— mediante *objetos*, compuestos por *unidades* codificadas o no, cada una de las cuales está conformada por una colección de *símbolos* pertenecientes a algún *campo finito* (§ 2.3.2). A continuación nos referimos con mayor detalle a cada una de esas etapas.

3.3.1. Procesamiento

En esta etapa la información se transforma para redundarse. Dicha transformación involucra hacer pasar la información por una serie de procedimientos que la codifican o multiplican en función de la disponibilidad pretendida para esa información; esos procedimientos son los que llamamos *Estrategias de Redundancia de Información* (§ 3.4).

El procesamiento de la información para redundarla es, normalmente, un trabajo exclusivamente centralizado y que supone la utilización, en buena medida, de ciclos de CPU —para la codificación y/o manipulación del la información— y espacio de almacenamiento en disco —temporal o permanentemente, según el diseño del sistema P2P en particular—. Las subetapas del procesamiento son (Fig. 3.8):

Figura 3.8: *Etapa de procesamiento de un sistema de almacenamiento P2P.*



P-1: *Adquisición*. El nodo donde se realizará el procesamiento adquiere una copia íntegra de cada objeto que se desea almacenar.

P-2: *Redundancia*. Se asume que el objeto inicial está formado de m unidades que se redundan hasta producir n nuevas unidades, usando una técnica predeterminada. El resultado de esta operación se expresa en términos de un factor k, que suele llamarse factor de estiramiento, tal que

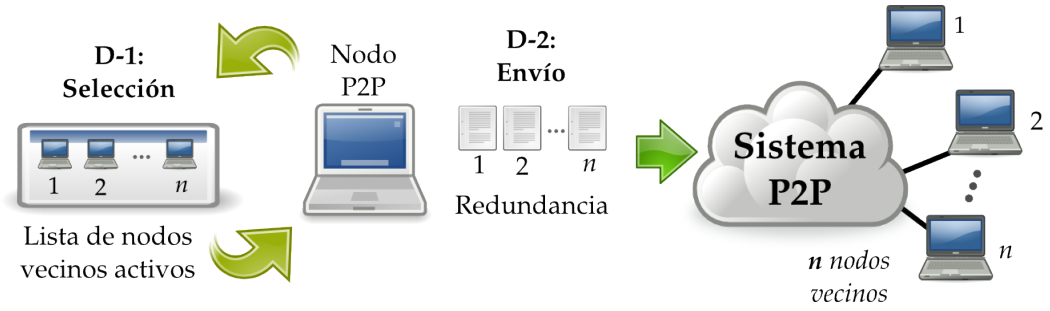
$$k = \frac{\text{Cantidad de información total después de la redundancia}}{\text{Cantidad de información antes de la redundancia}}.$$

3.3.2. Distribución

Una vez procesada la información, esta se envía a una serie de nodos del sistema de almacenamiento P2P, previamente seleccionados de acuerdo a las reglas del sistema P2P en uso. En esta etapa se realiza un «copiado» de las n unidades de información creadas en la etapa de anterior hacia otros nodos en el sistema

de almacenamiento. Esta transmisión de información puede ser punto-a-punto —e.g., *unicast*— o punto-multipunto —e.g., *multicast*—. Por supuesto, esto implica el uso de una infraestructura de comunicaciones apropiada —e.g., Internet— y de espacio suficiente en disco en los nodos destino dentro del sistema. La distribución de las unidades de información incluye las siguientes subetapas (Fig. 3.9):

Figura 3.9: *Etapa de distribución de un sistema de almacenamiento P2P.*



D-1: *Selección*. El nodo donde se crea la redundancia selecciona un grupo de n nodos activos en el sistema de almacenamiento, e intenta establecer una conexión con ellos.

D-2: *Envío y almacenamiento*. De las n unidades de información que produjo, el nodo a cargo del procesamiento envía una unidad *distinta* a cada uno de los nodos con los que se enlazó en el paso anterior.

3.3.3. Mantenimiento

Un sistema de almacenamiento P2P, además de su capacidad para utilizar el espacio de almacenamiento compartido por sus miembros, debe proveer mecanismos para *preservar* la redundancia de información generada a partir de estrategias como las mencionadas en § 3.4. Sin un adecuado *mantenimiento* de la redundancia, la información corre el riesgo de degradar su disponibilidad.

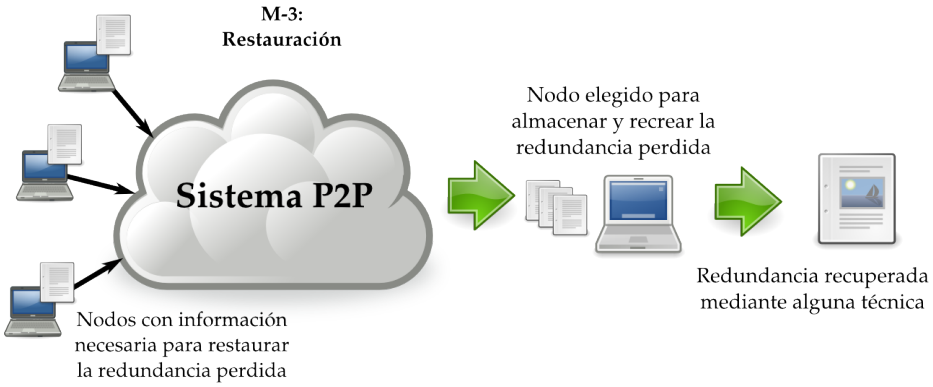
El mantenimiento de la redundancia de información en un sistema P2P es un tarea compleja que se ve afectada, principalmente, por la dinamicidad de los nodos. Esto es porque mientras más nodos entren y salgan del sistema, más tareas de restablecimiento de la información deben realizarse. Cada vez que un nodo parte, la información que almacenaba debe recuperarse mediante algún mecanismo, y una vez recuperada la información, ésta debe entregarse a otro miembro activo del sistema. En este sentido, un sistema P2P en condición de *churn* puede representar una verdadera amenaza para el rendimiento del sistema de almacenamiento. Esta etapa la desglosamos en tres subetapas, que son (Figs. 3.10 y 3.11):

M-1: *Monitoreo*. El sistema se vale de algún mecanismo diseñado para detectar e informar sobre la pérdida de redundancia en el sistema. El trabajo de monitoreo incluye el despliegue de software que registre los eventos ocurridos en los nodos, la adquisición de los datos que ellos generan —e.g., si el nodo está activo o no—, el procesamiento de esos datos y su entrega al sistema que lo solicita. El sistema P2P gobierna el sistema de monitoreo y le advierte cuando nuevos nodos deben ser monitoreados.

En la práctica, se utilizan heurísticas —e.g., membresías que expiran en el tiempo, como τ en la Fig. 3.11— para decidir cuándo ha partido un nodo del sistema de almacenamiento. Cuando el sistema de monitoreo estima que un nodo partió *definitivamente*, *cualquiera que sea la causa*, asume que hay pérdida de información que debe reponerse. En este trabajo de investigación llamaremos *fallas* a esas partidas definitivas.

Figura 3.10: *Etapa de mantenimiento de un sistema de almacenamiento P2P:* elección *y* monitoreo.

Figura 3.11: *Etapa de mantenimiento de un sistema de almacenamiento P2P:* restauración.



M-2: *Elección*. Aquí el sistema *elige* a qué nodo o grupo de nodos enviará una copia de la redundancia perdida. La elección se realiza en función de la probabilidad de que los nodos se encuentren activos en el sistema, i.e., en función de su disponibilidad.

M-3: *Restauración*. El nodo que restaura la información busca y se conecta a un subconjunto de nodos del sistema de almacenamiento que albergan las piezas de información necesarias para su labor. La información perdida se reconstruye de acuerdo a la técnica de redundancia utilizada en la etapa de procesamiento. El nodo que realiza esta tarea también está predeterminado por el diseño mismo del sistema; normalmente, será el mismo que fue elegido en M-2.

De las etapas mostradas, la subetapa M-3 es particularmente importante para los sistemas de almacenamiento. La razón es que, en sistemas P2P *reales*, durante esta etapa se utiliza un recurso que puede ser el más escaso de todos: el *ancho de banda*. Más aún, en M-3 se establecen las diferencias entre las estrategias de redundancia de información cuando son aplicadas a un sistema de almacenamiento P2P. En otras palabras, *todas* las demás subetapas pueden considerarse genéricas y válidas para cualquier sistema de almacenamiento distribuido y técnicas de redundancia de información. Las diferencias que existen en la subetapa M-3 estriban en la manera en que se recaba la información para volver a generar la que se ha dado por perdida; cada técnica de redundancia requiere de un proceso diferente para dicha tarea. En § 3.4 nos referiremos con mayor detalle en la subetapa M-3 de cada técnica de redundancia de información.

3.4. Estrategias de redundancia de información

Es un gran reto tecnológico construir sistemas P2P de alta disponibilidad, especialmente cuando estos se conforman por millones de equipos de cómputo administrados individualmente, que pueden entrar o salir del sistema de manera arbitraria y donde este comportamiento es una condición normal y permanente.

En la práctica y en la literatura existen diversas estrategias de redundancia de información que buscan asegurar altos niveles de disponibilidad de información bajo sistemas de almacenamiento P2P a cambio de ciertos costos en el uso de recursos de cómputo —i.e., espacio de almacenamiento, ancho de banda de la red de comunicaciones y ciclos de CPU.

En las subsecciones siguientes clasificaremos cada estrategia dentro de tres grupos que reflejan el tratamiento que le dan a la información original *antes* de ser distribuida en el sistema. Dicha clasificación considera lo que sucede durante la etapa de procesamiento del almacenamiento P2P (§ 3.3). También hacemos mención de la subetapa de restauración M-3 particular de cada estrategia de redundancia, e incluimos diagramas genéricos de cómo se utilizarían esas estrategias en un sistema de almacenamiento P2P. La evaluación y análisis de cada estrategia, en función de su impacto en diversos parámetros de desempeño, como la disponibilidad de la información, se encuentra en la Parte II de este documento.

3.4.1. Estrategias sin códigos

Un objeto de información puede multiplicarse en el sistema de almacenamiento distribuyendo copias de sí mismo, ya sea de manera íntegra, o de cada una de las unidades en las que llegue a dividirse. A la primera forma le llamaremos redundancia simple; a la segunda, redundancia por bloques. La cantidad de copias creadas deberá ajustarse apropiadamente para alcanzar los niveles de disponibilidad deseados para un objeto de información dado.

- **Redundancia simple** (RS)

  Es el esquema más sencillo de redundancia de información, donde n copias de cada objeto de información son distribuidas entre n miembros del sistema. CAN y PAST son sistemas de almacenamiento P2P que utilizan redundancia simple (ver Fig. 3.12).

- **Redundancia por bloques** (RB)

  Bajo este esquema, dividimos un objeto de información en m unidades o bloques y hacemos n copias de cada uno. De esta manera, obtenemos $m \times n$ unidades de información que debemos distribuir en la misma cantidad de nodos en el sistema de almacenamiento. CFS [22] y eDonkey se valen de esta estrategia para distribuir y multiplicar la información de disponible en ellos (ver Fig. 3.13).

3.4.2. Estrategias con códigos

Las estrategias de redundancia de información que aquí mostramos utilizan la técnicas de codificación revisadas en § 2.3.2:

- **Redundancia con IDA** (RIDA, Fig. 3.14)
- **Redundancia con códigos de red** (RCR, Fig. 3.16)

  Cabe mencionar que la Fig. 3.16 también muestra cómo se transmiten los vectores de coeficientes. Esto es sólo para efectos ilustrativos, pero que no se tomarán en cuenta para las evaluaciones porque consideramos que su efecto puede despreciarse debido a su tamaño. Todas las estrategias con códigos también utilizan vectores de coeficientes anexos a los dispersos durante su distribución.

- **Redundancia con fuente digital** (RFD, Fig. 3.15)

3.4.3. Estrategia híbrida

En esta estrategia de redundancia (RH), además de distribuir unidades codificadas de información como se muestra en la Fig. 3.14, mantenemos una copia del objeto original dentro del sistema. Esto significa que en el sistema de almacenamiento P2P hay n unidades más un objeto de información que pueden utilizarse para recuperar redundancia perdida. Es decir, RH = RIDA+RS (1 copia). Así, cada vez que una unidad se pierda no será necesario reunir las demás para reconstruir el objeto original, como sucede con la redundancia mediante IDA. Este esquema de redundancia de información fue concebido originalmente en [12].

Figura 3.12: **Redundancia simple**. **(1)** *En algún nodo se recibe para almacenamiento un objeto F* (Etapa P-1), *el cual luego se duplica en* n *copias íntegras* (Etapa P-2). **(2)** *El nodo que actúa como fuente de información selecciona* n *nodos* (Etapa D-1), *para almacenar en ellos cada una de las copias de F creadas* (Etapa D-2). **(3)** *Si el sistema detecta la pérdida de alguna de las copias* (Etapa M-1), *elige un nuevo nodo —e.g., un recién llegado— para enviarle* (Etapa M-2), *y así restaurar, la copia faltante* (Etapa M-3). *El procedimiento anterior también aplica cuando algún nodo solicita una copia de la información original.*

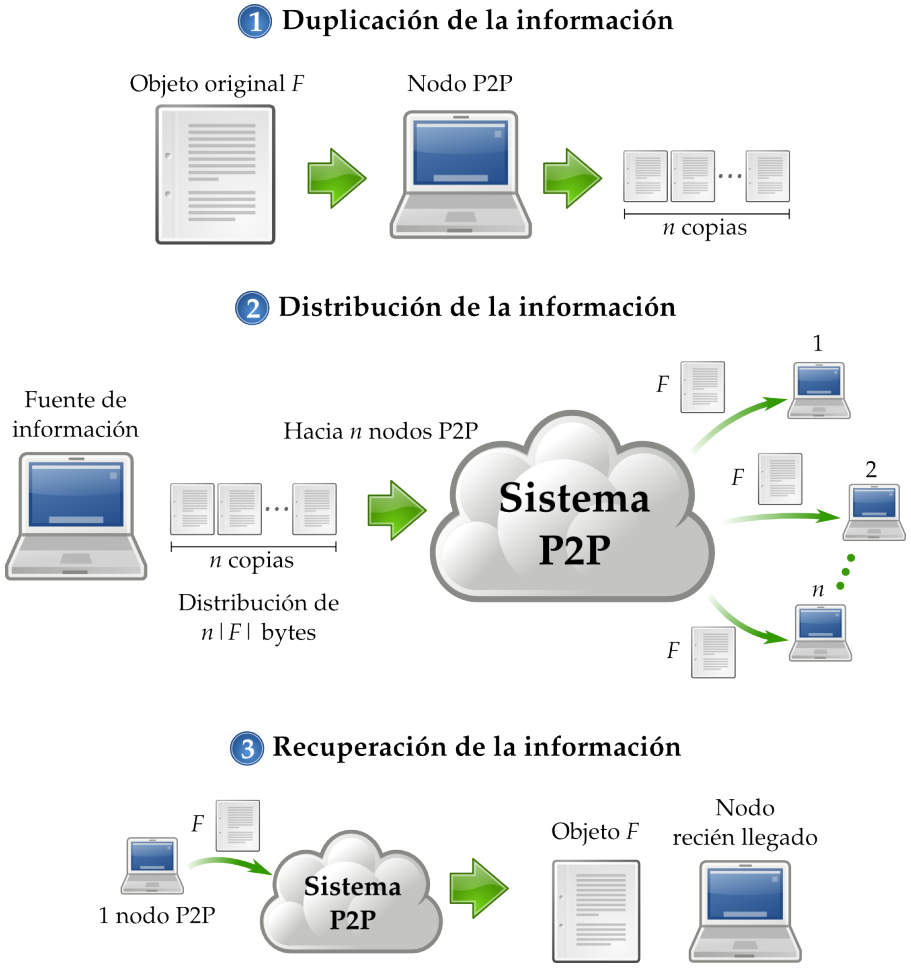


Figura 3.13: **Redundancia con bloques**. **(1)** *En algún nodo se recibe para almacenamiento un objeto F* (Etapa P-1), *éste se divide en* n *unidades de igual tamaño, las cuales se duplican en* m *copias cada una* (Etapa P-2). **(2)** *El nodo que actúa como fuente de información selecciona* nm *nodos* (Etapa D-1), *para almacenar en ellos cada uno de los* nm *bloques* $F_{i,j}$ *creados* (Etapa D-2). **(3)** *Si el sistema detecta la pérdida de alguno de los bloques* (Etapa M-1), *elige un nuevo nodo para enviarle* (Etapa M-2), *y así restaurar, el bloque faltante* (Etapa M-3). **(4)** *Si hay una solicitud para obtener la información original, el sistema envía* n *copias de bloques diferentes al nodo solicitante para que éste las una apropiadamente para reconstruir F*.

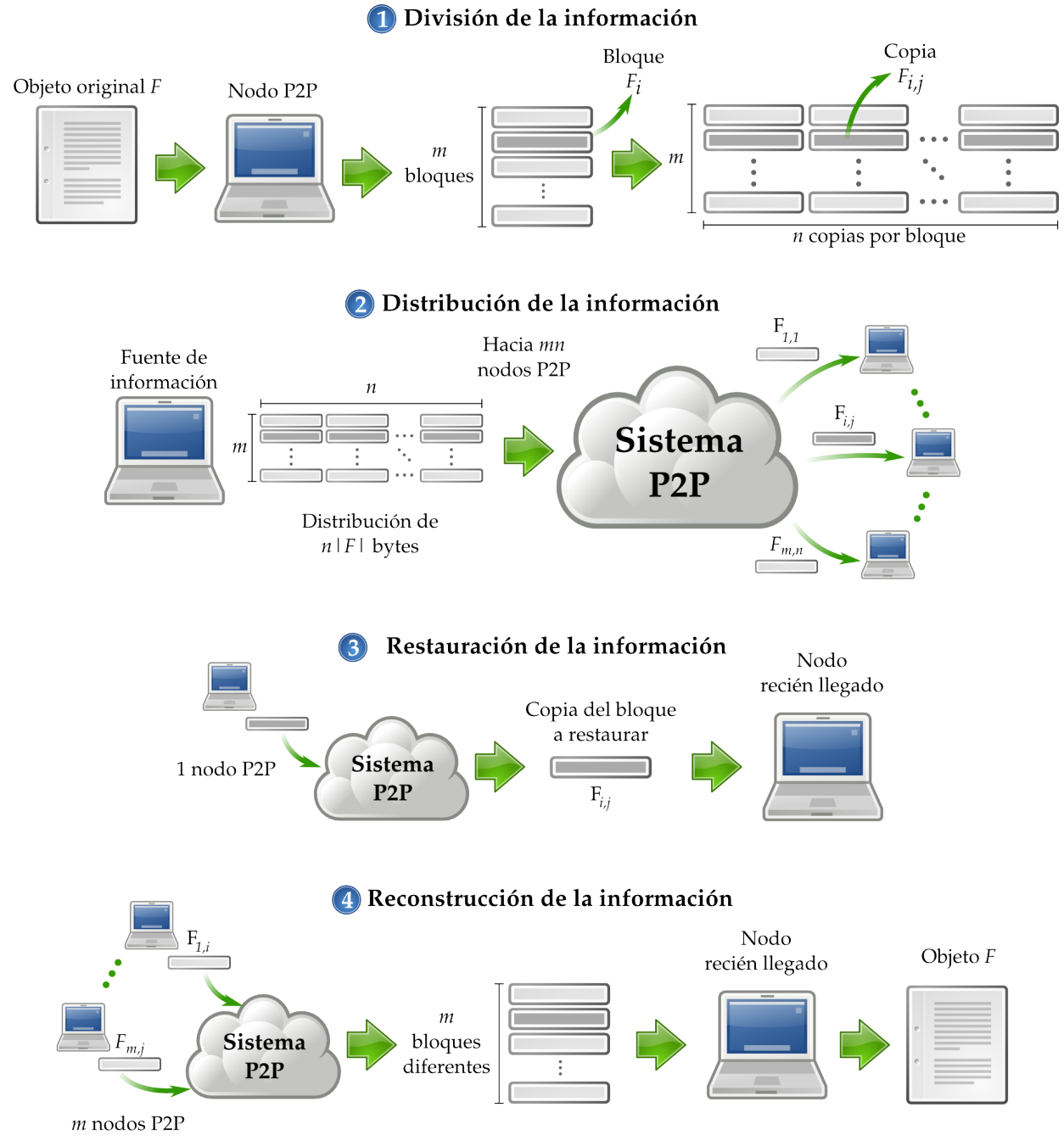


Figura 3.14: **Redundancia con IDA.** **(1)** *En algún nodo se recibe para almacenamiento un objeto F* (Etapa P-1) *que se considera formado por* N *símbolos, cada uno de los cuales son elementos del campo* $\mathbb{F}_p$. *El objeto se divide en vectores o unidades de* m *símbolos cada uno.* **(2)** *El conjunto de vectores pasa por una transformación lineal que da lugar a* n *unidades codificadas a las que llamamos dispersos* (Etapa P-2). **(3)** *El nodo que actúa como fuente de información selecciona* n *nodos* (Etapa D-1) *para almacenar en ellos cada uno de los* n *dispersos resultantes* (Etapa D-2). **(4)** *Si hay una solicitud para obtener el objeto original, el sistema envía* m *de* n *dispersos diferentes al nodo solicitante para que éste los decodifique —con la matriz* $\hat{\mathbf{A}}$ *asociada con la transformación lineal— para reconstruir F. Si el sistema detecta la pérdida de alguno de los dispersos* (Etapa M-1), *elige un nodo recién llegado* (Etapa M-2), *sigue el paso* **(4)** *para reconstruir allí la información original, y luego el paso* **(2)** *para restaurar el disperso faltante* (Etapa M-3).

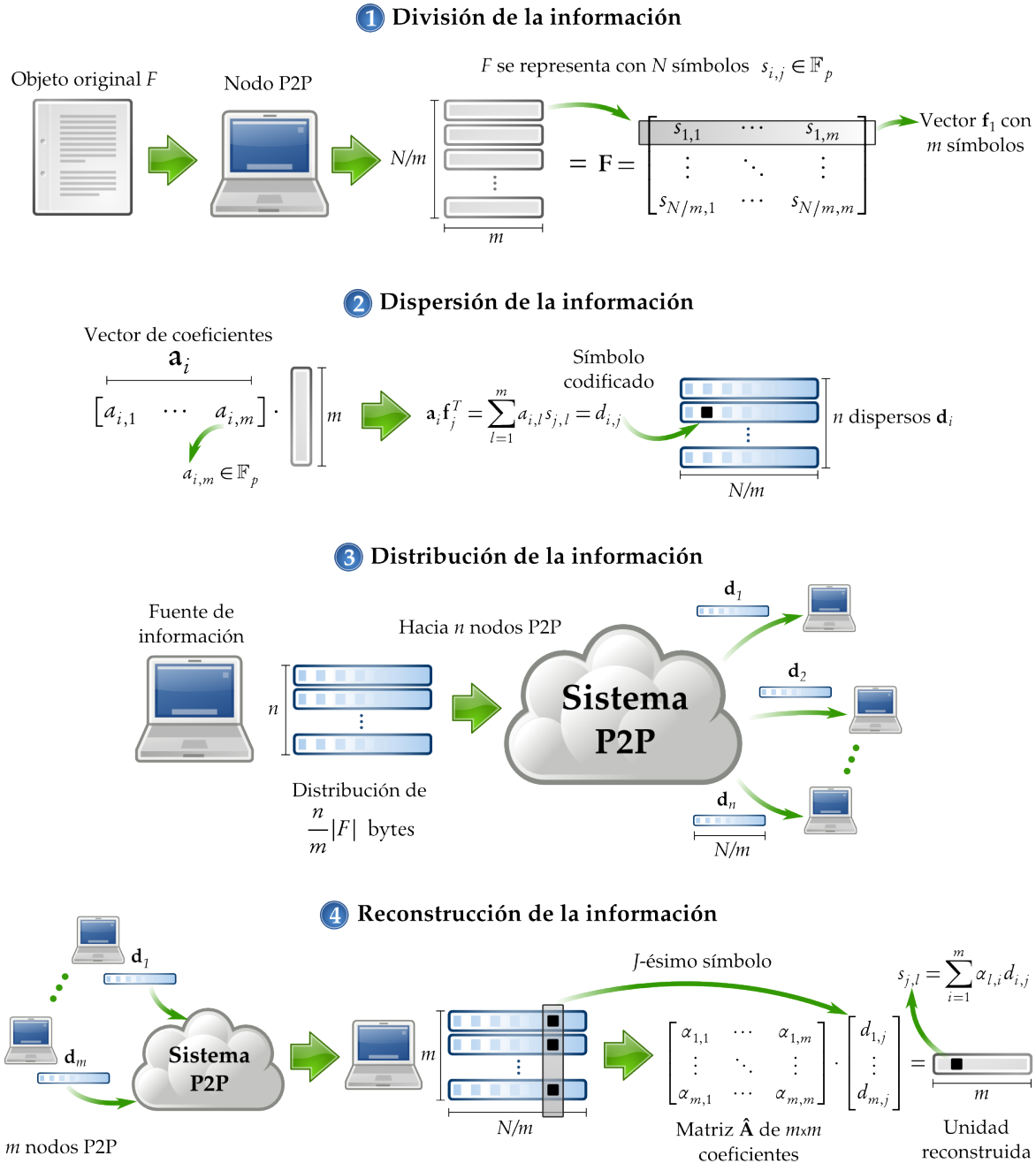


Figura 3.15: **Redundancia con fuente digital**. **(1)** *En algún nodo se recibe para almacenamiento un objeto* F (Etapa P-1) *que se considera formado por* N *símbolos, cada uno de los cuales son elementos del campo* $\mathbb{F}_2$ *—i.e., bits—. El objeto se divide en vectores o unidades de* m *símbolos cada uno.* **(2)** *El objeto original se procesa con un vector de* N/m *coeficientes generado "al vuelo" para crear, a través de una combinación lineal, unidades codificadas a las que llamamos dispersos* (Etapa P-2). **(3)** *El nodo que actúa como fuente de información —en este caso, una fuente digital— selecciona nodos* (Etapa D-1), *tantos como sean necesario,* $n > N/m$, *para almacenar en ellos cada uno de los dispersos que se van generando* (Etapa D-2). **(4)** *Si hay una solicitud para obtener el objeto original, el sistema envía* $N/m+\epsilon$ *copias de dispersos diferentes al nodo solicitante, éste toma* N/m *de ellos, y los decodifica —con la matriz* $\hat{\mathbf{A}}$ *de coeficientes asociada— para reconstruir* F *con una probabilidad* $\delta(\epsilon) \leq 2^{-\epsilon}$ *(§ 2.3.2). Si el sistema detecta la pérdida de alguno de los dispersos* (Etapa M-1), *elige un nodo recién llegado para que solicite a la fuente de información que le distribuya* (Etapa M-2), *como sucede en el paso* **(4)**, *un nuevo disperso —cualesquiera— generado como en el paso* **(2)**. *Así el nodo recién llegado almacenará un disperso para restaurar la redundancia perdida* (Etapa M-3).

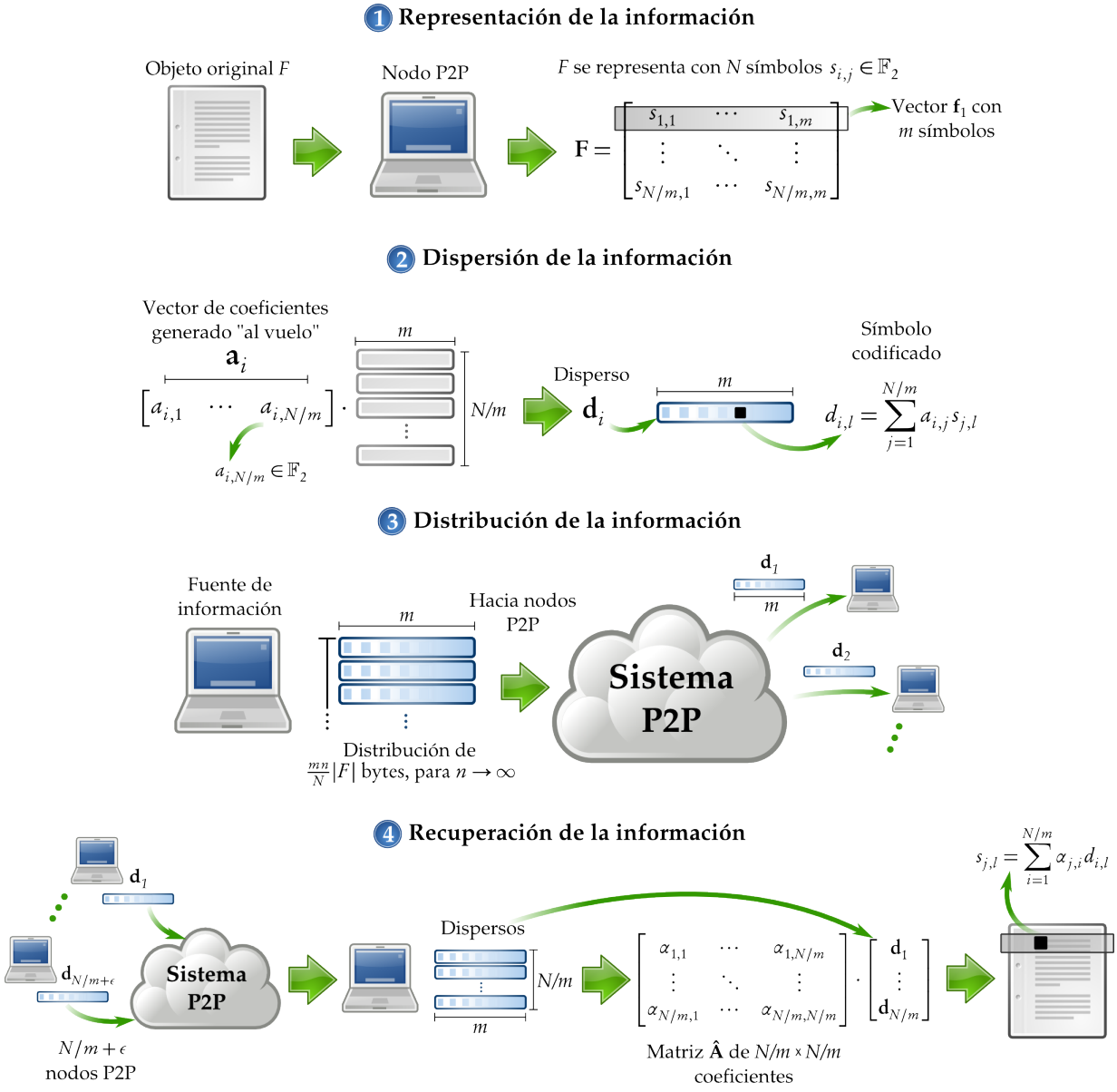


Figura 3.16: **Redundancia con códigos de red**. **(1)** *En algún nodo se recibe para almacenamiento un objeto F* (Etapa P-1) *que se considera formado por* N *símbolos, cada uno de los cuales son elementos del campo* $\mathbb{F}_p$. *El objeto se divide en vectores o unidades de* m *símbolos cada uno.* **(2)** *El objeto original se procesa con un vector de* N/m *coeficientes generado "al vuelo" para crear, a través de una combinación lineal aleatoria, unidades codificadas a las que llamamos dispersos* (Etapa P-2). **(3)** *El nodo que actúa como fuente de información selecciona nodos* (Etapa D-1) *para almacenar en ellos cada uno de los dispersos generados, junto con el vector de coeficientes asociado* (Etapa D-2). **(4)** *Los dispersos se pueden recombinar en cualquier nodo para generar nuevos dispersos, mismos que reexpide posteriormente.* **(5)** *Si hay una solicitud para obtener el objeto original, el sistema envía N/m copias de dispersos diferentes al nodo solicitante y los decodifica —con la matriz* $\hat{\mathbf{A}}$ *de coeficientes asociada— para reconstruir F. Si el sistema detecta la pérdida de alguno de los dispersos* (Etapa M-1), *elige un nodo recién llegado para que solicite a la fuente de información* (Etapa M-2), *siguiendo el paso* **(3)**, *un nuevo disperso creado como se muestra en el paso* **(2)**, *y restaurar así la redundancia perdida* (Etapa M-3). *Asimismo, el nodo recién llegado puede seguir el paso* **(4)** *para recombinar dispersos y obtener uno nuevo.*

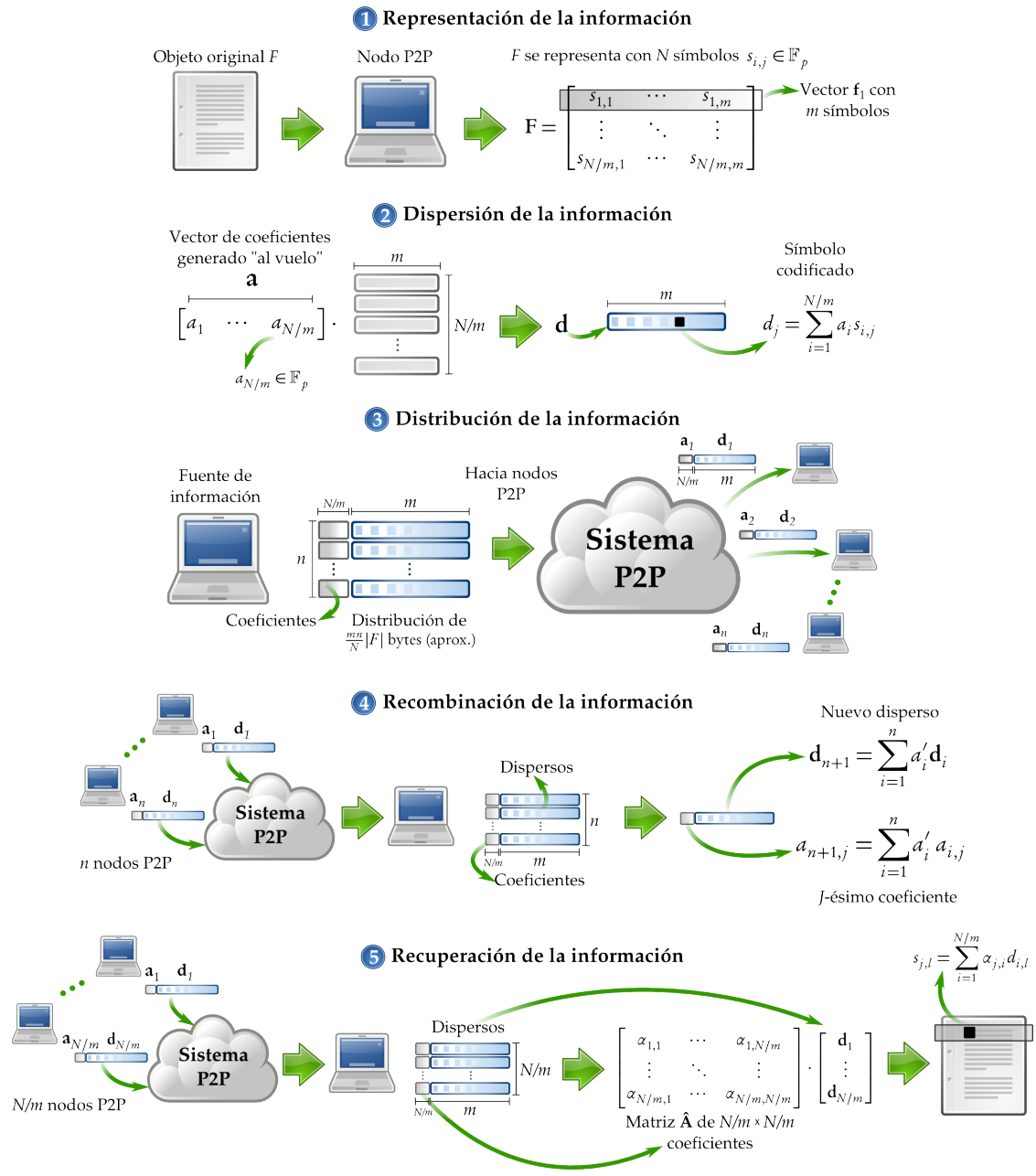
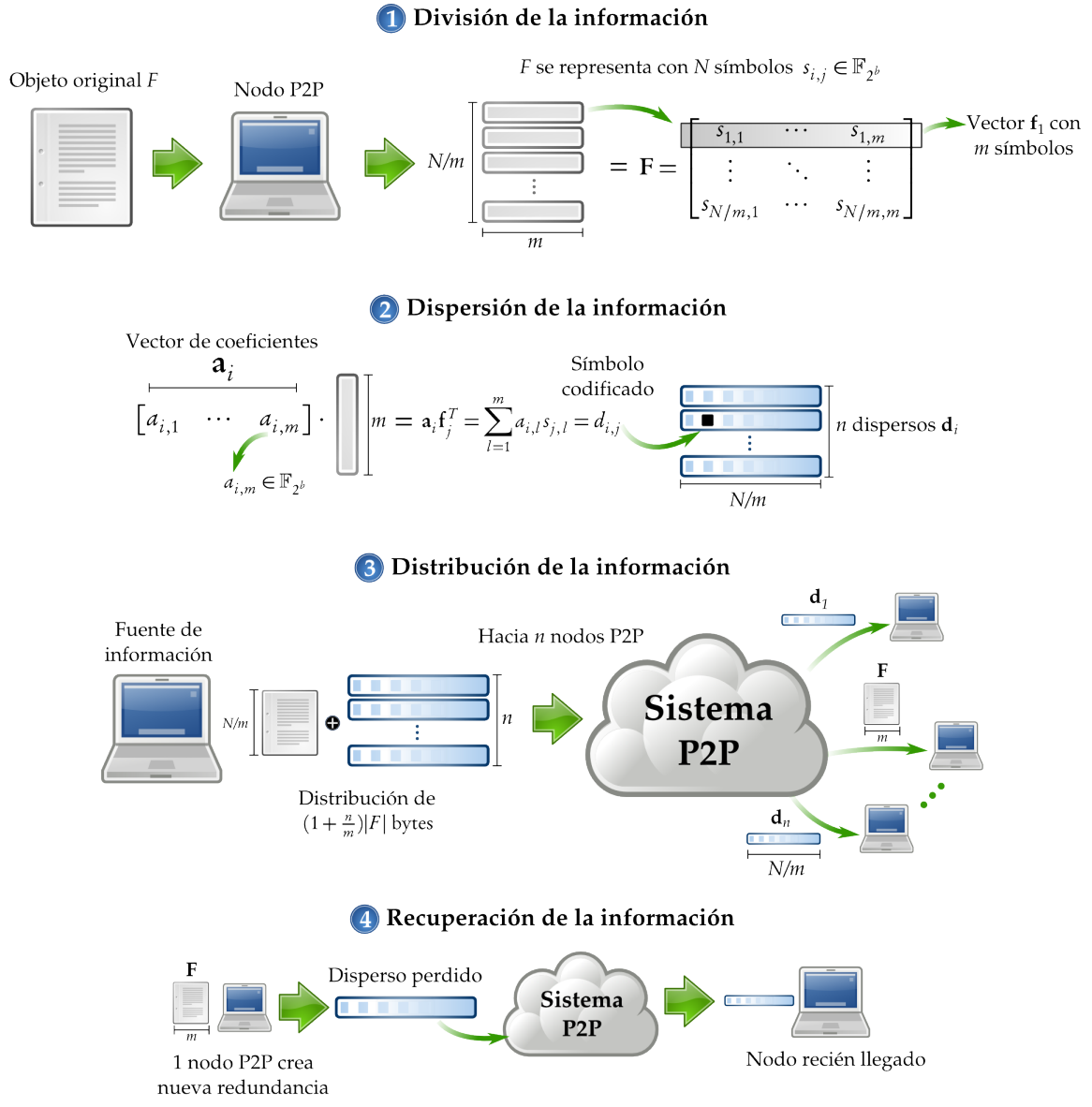


Figura 3.17: **Redundancia con estrategia híbrida: RIDA + RS (1 copia). (1)** *En algún nodo se recibe para almacenamiento un objeto F* (Etapa P-1) *que se considera formado por* N *símbolos, cada uno de los cuales son elementos del campo* $\mathbb{F}_p$. *El objeto se divide en vectores o unidades de* m *símbolos cada uno.* **(2)** *El conjunto de vectores pasa por una transformación lineal que da lugar a* n *unidades codificadas a las que llamamos dispersos* (Etapa P-2). **(3)** *El nodo que actúa como fuente de información selecciona* n+1 *nodos* (Etapa D-1) *para almacenar en ellos cada uno de los* n *dispersos resultantes más una copia íntegra del objeto original* (Etapa D-2). **(4)** *Si el nodo que tiene una copia de F está disponible y el sistema detecta la pérdida de algún disperso* (Etapa M-1), *entonces el disperso perdido puede reconstruirse a partir de F sin necesidad de reunir más dispersos* (Etapa M-3), *y enviarlo al nodo elegido para almacerlo* (Etapa M-2). *Y F, cuando sea requerido, puede recuperarse como se hace con la redundancia simple: enviando una copia. De lo contrario, con F perdido, entonces tenemos que sólo es posible usar IDA, por lo que las operaciones de restauración y recuperación de información se deben realizar como se indica en la Fig. 3.14.*



Parte II

Evaluación

Capítulo 4

Trabajo Previo

En esta sección examinaremos una serie de trabajos representativos en relación con la evaluación de técnicas de duplicación para sistemas de almacenamiento P2P. En particular, categorizamos los avances de [91, 9, 75] y [23] en función del comportamiento —estático o dinámico— de los nodos del sistema P2P que asumieron para realizar sus análisis. Al final de este capítulo mostramos la Tabla 4.1, que sintetiza nuestra revisión de esos trabajos.

4.1. Evaluación con modelos estáticos

Los trabajos de [91] y [9] se caracterizan por evaluar técnicas de duplicación bajo modelos estáticos de sistemas P2P. En el estudio de [91] formularon un modelo matemático para cuantificar valores $MTTF$ (§ 2.1.1), ancho de banda y almacenamiento de sistemas P2P que usan redundancia simple, o bien, redundancia con códigos de borrado tipo MDS, como el IDA —RS y RIDA, respectivamente—. También asumieron que las fallas en los discos de los nodos que componen el sistema son independientes e idénticamente distribuidas —i.i.d.—, pero sin considerar aspectos dinámicos de los sistemas P2P. En su trabajo concluyeron que la estrategia RIDA provee valores de $MTTF$ de varios órdenes de magnitud más altos en comparación con la estrategia RS, para la misma redundancia de información; además, los RIDA proveen un orden de magnitud menor de uso de ancho de banda y almacenamiento frente a RS para los mismos valores de $MTTF$.

Por su parte, en [9] formularon un modelo probabilístico para razonar sobre el exceso de almacenamiento requerido para entregar niveles específicos de disponibilidad de la información y en función de la disponibilidad de los nodos. Dicho modelo fue validado mediante simulaciones de tipo Monte-Carlo para cierta cantidad de nodos, tamaño de archivos y cierta disponibilidad media de los nodos, asumiendo también que las fallas se daban i.i.d. Básicamente, simularon un conjunto de nodos que podían estar disponibles o no, en el que distribuyeron archivos mediante RS o RIDA de acuerdo con un modelo probabilístico, para luego medir el impacto en almacenamiento promedio por nodo. También estudiaron los efectos a nivel sistema —i.e., en todos los nodos— de la disponibilidad y el almacenamiento. Bajo ese estudio, RIDA ofrece una importante ventaja frente a RS: *requiere menos espacio de almacenamiento para ofrecer los mismos niveles de disponibilidad de la información*.

Además, en [9] hicieron un primer intento por mostrar los efectos de un sistema P2P dinámico con la inclusión de un análisis de la disponibilidad a largo plazo, y donde asumieron tasas de abandono de usuarios de acuerdo a una distribución exponencial. Ellos encontraron que la estrategia RIDA, a lo largo del tiempo, puede llevar a una rápida degradación de la disponibilidad de la información en la medida que los nodos parten del sistema.

En [2], por otro lado, consideraron el problema de almacenar múltiples objetos de información en múltiples nodos de almacenamiento. Donde cada nodo recibe para almacenar una o más unidades, codificadas o

no, del objeto original. Ellos compararon el desempeño de tres estrategias: RS, RIDA y los códigos lineales aleatorios (§ 2.3.2) que son utilizados por los códigos de red —RCR—. Para sus evaluaciones crearon un modelo probabilístico, donde un objeto se divide en m unidades, cada nodo puede almacenar l copias de esas m, y para reconstruir la información original, el nodo se conecta a un número fijo r de nodos. En [2] concluyen que los códigos de red pueden considerarse como *códigos de borrado al vuelo*.

4.2. Evaluación con modelos dinámicos

Los trabajos de [12, 75] y [23] se caracterizan por evaluar técnicas de duplicación bajo modelos dinámicos de sistemas P2P. Les llamamos dinámicos porque toman en cuenta el cambio de estado de los nodos P2P a lo largo del tiempo. En este sentido, si bien es posible emular el comportamiento de un sistema P2P, es de práctica común utilizar los datos generados por sistemas P2P existentes —e.g., tiempo de entrada y salida de cada nodo— para lograr resultados más cercanos a la realidad. A dichos datos se les llama *trazas*, y pueden servir como datos de entrada en simulaciones de los sistemas que representan.

En [12, 75] formularon un modelo matemático donde la redundancia requerida por RS y RIDA está en función de la disponibilidad requerida por objeto de información. Asimismo, elaboraron un modelo de mantenimiento de información basado en la tasa λ de entrada y la tasa μ de salida de nodos y el consecuente impacto en la trasferencia de información, bajo los supuestos de que $\lambda = \mu$ y la cantidad de nodos en el sistema permanece constante.

En [12, 75] también definieron la heurística τ, el *tiempo de expiración de membresía* de un nodo en el sistema, para poder distinguir entre partidas temporales y permanentes. Con base en esa distinción obtuvieron una fórmula para estimar el uso de ancho de banda promedio por nodo en función de τ. Ellos evaluaron su modelo procesando trazas de tres sistemas P2P con diferentes características entre sí: Overnet [10], Farsite [14] y PlanetLab [87]. Cabe mencionar que ellos consideraron en su análisis el uso de una copia completa de cada objeto de información en el sistema, en adición a la redundancia provista mediante RIDA, dando lugar al esquema de redundancia híbrido —RH— mencionado en § 3.4.3.

A diferencia de los trabajos donde se estudiaron modelos estáticos, en [75] no concluyeron que RIDA es la mejor estrategia de redundancia de información para todos los casos, sino que acotaron su éxito a ambientes donde la disponibilidad promedio de los nodos se ubica entre el 80 y 90 %, como sucede en Farsite. Los códigos de borrado tipo IDA agregan complejidad innecesaria en ambientes tan estables como el de PlanetLab, donde la estrategia RS es más adecuada, y acarrean una complejidad excesiva en su administración, sobre todo en ambientes tan inestables como el de Overnet. En [12], afirmaron que *no es posible conseguir alta disponibilidad de la información, uso eficiente del almacenamiento, y sistemas P2P dinámicos, sin afectar negativamente alguna de las tres*. Los resultados de [75] y [12] fueron confirmados y complementados analíticamente por [46].

Por otro lado, en [23] fueron más lejos que sus predecesores. Evaluaron, partiendo del modelo creado por [75], estrategia RS, RIDA bajo dos situaciones: 1) una estrategia RIDA ideal, donde nuevos dispersos son creados sin necesidad de transferir información para ello; 2) una estrategia RH [75]; 3) una estrategia que llaman OMMDS (§ 2.3.2). También evaluaron una estrategia tipo RCR que llaman *códigos de regeneración* —para estos dos últimos véase también § 2.3.3.

Los resultados de [23] se obtuvieron a partir simulaciones basadas en cuatro trazas: PlanetLab, Farsite, Gnutella [37] y Skype [36]. De las cuales concluyeron que los códigos de regeneración proveen mayores beneficios que las otras estrategias redundancia en cuanto el uso de ancho de banda, en particular, si el sistema de almacenamiento es para respaldar información con archivos grandes. Sin embargo, la estrategia RH es muy eficiente en la mayoría de los casos.

Tabla 4.1: *Síntesis —ordenada cronológicamente— de trabajos relacionados con la evaluación de estrategias de redundancia de información en sistemas de almacenamiento P2P.*

Artículo:	*Erasure coding vs. replication: A quantitative comparison* [91].
Año:	2002.
Metodología:	Modelación probabilística de la disponibilidad de la información.
Estrategias:	RS y RIDA.
Etapas:	P-2 y D-2.
Parámetros:	$MTTF$ de las unidades de almacenamiento de los nodos P2P. Almacenamiento y ancho de banda —de lectura y escritura— de reparación.
Aportación:	Modelo probabilístico del $MTTF$ para una estrategia RIDA.
Conclusiones:	RIDA requiere una orden de magnitud menor de almacenamiento que RS.
Artículo:	*Replication strategies for highly available peer-to-peer storage* [9].
Año:	2003
Metodología:	Modelación probabilística. Simulación con el método de Monte-Carlo.
Estrategias:	RS, RB y RIDA.
Etapas:	P-2 y D-2.
Parámetros:	Almacenamiento y disponibilidad de la información; estática y a largo plazo.
Aportación:	Modelo probabilístico de la disponibilidad de la información basado en la disponibilidad de los nodos del sistema.
Conclusiones:	RS requiere mucho menos almacenamiento para ofrecer la misma disponibilidad de la información.
Artículo:	*High availability, scalable storage, dynamic peer networks: pick two* [12].
Año:	2003.
Metodología:	Modelo probabilístico de [9]. Recolección de trazas de la red Gnutella.
Estrategias:	RS y RIDA.
Etapas:	P-2, D-2 y M-3.
Parámetros:	Almacenamiento, ancho de banda y disponibilidad de la información.
Aportación:	Modelo de mantenimiento de la redundancia para un sistema de almacenamiento P2P. Utilización de trazas reales.
Conclusiones:	Hay un compromiso entre alta disponibilidad de la información, el uso eficiente del almacenamiento y el ancho de banda de mantenimiento.
Artículo:	*High Availability in DHTs: Erasure Coding vs. Replication* [75].
Año:	2005.
Metodología:	Modelo probabilístico [9]. Modelo de mantenimiento de [12]. Simulación basada en trazas de sistemas P2P.
Estrategias:	RS y RIDA.
Etapas:	P-2, D-2 y M-3.
Parámetros:	Almacenamiento, ancho de banda y disponibilidad de la información.
Aportación:	Panorama más completo del desempeño de las técnicas de duplicación estudiadas. Creación de una técnica de duplicación híbrida.
Conclusiones:	El desempeño de la estrategia híbrida es mejor cuando la disponibilidad promedio de los nodos está entre el 80 y 90 %

— Continúa en la siguiente página. —

Tabla 4.1: *Síntesis de trabajos relacionados.* —Continuación.—

Artículo:	*How good is random linear coding based distributed networked storage?* [2].
Año:	2005.
Metodología:	Modelo probabilístico.
Estrategias:	RS, RIDA y RCR.
Etapas:	P-2 y D-2.
Parámetros:	Disponibilidad de la información y almacenamiento.
Aportación:	Modelo probabilístico parametrizable en tres sentidos: r, nodos a los que se descarga para la reconstrucción; m, unidades en las que se divide el objeto original; l, unidades almacenadas por nodo.
Conclusiones:	Los códigos de red pueden proveer mayores niveles de disponibilidad sin afectar el almacenamiento.
Artículo:	*Network Coding for Distributed Storage Systems* [23].
Año:	2007.
Metodología:	Modelo probabilístico [9] y de mantenimiento de [12]. Simulación basada en trazas de sistemas P2P.
Estrategias:	RS, RIDA, RH y RCR.
Etapas:	P-2, D-2 y M-3.
Parámetros:	Ancho de banda y disponibilidad de la información.
Aportación:	Uso de códigos de red para mejorar el desempeño de los códigos de borrado en la duplicación de información.
Conclusiones:	Los códigos de red pueden reducir significativamente el ancho de banda de mantenimiento.

4.3. Discusión

En § 4.1 y 4.2 revisamos trabajos de investigación que consideramos representativos de la investigación sobre cómo distribuir, almacenar y restaurar información en sistemas de almacenamiento P2P. Apreciamos en ellos que

- Hay una evolución en la representación de los sistemas P2P, pasando de los que llamamos estáticos —estudian el estado del sistema en un instante de tiempo— a los dinámicos —estudian el sistema a lo largo del tiempo—.
- Todos se basan, principalmente, en el modelo probabilístico propuesto por [9].
- No hay un trabajo que compendie, a modo de vista panorámica y de manera coherente, los resultados conseguidos hasta el momento. Aunque, por otro lado, cada trabajo representa un hito para la investigación en el tema.
- No existe una nomenclatura estándar, i.e., cada trabajo usa la propia.

Entonces, luego de esa revisión, consideramos conveniente aportar lo siguiente:

- Realizar un compendio ordenado y detallado de los resultados más importantes encontrados en relación con la evaluación de estrategias de redundancia en sistemas de almacenamiento P2P.

- Agregar resultados experimentales que complementen y faciliten la comprensión del comportamiento de cada estrategia. Por ejemplo, al proponer nuevas gráficas a las existentes en la literatura.
- Proponer una nomenclatura adecuada para la temática.
- Concluir con la realización de una guía que intersecte todas las estrategias evaluadas.

Así intentaremos dar cumplimiento a nuestro objetivo de guiar al diseñador de un sistema de almacenamiento P2P en la elección de la estrategia más adecuada para redundar información en función de una serie de factores determinantes.

Capítulo 5

Metodología

Si bien en secciones anteriores (§ 3.4) mostramos qué estrategias son las más utilizadas para redundar información en sistemas de almacenamiento P2P, no hablamos de *cuánta* redundancia requiere cada una de ellas para lograr los niveles deseados de disponibilidad de la información, ni cuáles son los otros costos asociados con esa tarea para diferentes estados del sistema. Podemos encontrar la respuesta a esas preguntas en los trabajos de investigación descritos en § 4.

Por lo que respecta a los capítulos que siguen, en § 6, § 7 y § 8, presentaremos en detalle los experimentos y resultados de los trabajos citados con anterioridad. Para esto reproduciremos los experimentos originales; así podremos servirnos de la misma plataforma de pruebas y agregar otros resultados que consideremos complementarios para ampliar la comprensión del impacto de cada estrategia de redundancia de información. En este sentido, nuestra aportación consiste de

- Comparaciones entre más estrategias de redundancia.
- Simulaciones bajo un espectro más amplio de valores de los parámetros de desempeño.
- Cálculo de intervalos de confianza para cada resultado numérico obtenido a través de simulaciones.

Para lograr lo anterior, *evaluaremos* estrategias de redundancia de información en sistemas de almacenamiento P2P, de la siguiente manera:

- *Una definición para cada una de las medidas de desempeño elegidas*. Anteriormente, en § 2.1.1 y § 2.2.2 mencionamos diversas medidas de desempeño para sistemas de cómputo y de almacenamiento P2P, respectivamente. De entre todas ellas retomaremos las que nos hablan de la *disponibilidad* de la información, el *ancho de banda* requerido para mantener los niveles de redundancia de información, el espacio *almacenamiento* debido la redundancia de información.
- *Un modelo del sistema de almacenamiento asumido para realizar el análisis teórico y las simulaciones correspondientes*. Justo aquí conviene presentar el modelo probabilístico creado por [9] para determinar la disponibilidad de la información en un sistema de almacenamiento P2P. Esto es importante porque en todas las técnicas de redundancia tratadas subyace tal modelo, enmarcardo por estas suposiciones:
  - Suponemos un sistema de almacenamiento compuesto de $\mathcal{N}$ nodos, cada uno con una probabilidad media a de estar disponible, sobre los cuales duplicamos —con la estrategia elegida— un total de $\mathcal{F}$ diferentes archivos cuyo tamaño tiene una media de M bytes.
  - Las fallas en los nodos que conforman el sistema distribuido son i.i.d. Esto permite que las unidades de la información original puedan asignarse aleatoriamente a cada nodo. Experimentalmente, esta política de distribución de redundancia es relativamente simple de simular.

  - Los nodos son la mínima abstracción del sistema de almacenamiento que asumimos y no existen más elementos de falla; e.g., discos duros llenos, falta de energía eléctrica, fallas en la red. Asimismo, el modelo presupone que cada nodo provee una unidad de almacenamiento cuya capacidad es muy grande.

- *Una serie de resultados hallados en la literatura citada y los obtenidos en este trabajo de investigación.* Estos resultados pueden ser teóricos y prácticos; e.g., fórmulas, gráficas, modelos. Junto a estos, incluiremos nuevas gráficas y simulaciones —señaladas oportunamente—, para complementar y facilitar la comprensión de cada estrategia estudiada. Además, calculamos intervalos de confianza a los resultados obtenidos a través de las simulaciones tipo Monte-Carlo.

- *La evaluación del peor de los casos.* Esto es, cuando los nodos sólo almacenan una unidad de la información y estos sólo se conectan a un número mínimo posible de nodos para recuperar el objeto original. Sabemos que se trata del *peor* de los casos por los resultados del trabajo de [2].

- *Una comparación*, mediante gráficas, entre los resultados obtenidos.

- *Una propuesta de nomenclatura* que intentará ser clara y coherente a lo largo de cada evaluación.

Capítulo 6

Evaluación de la redundancia

En este capítulo realizamos un estudio de la Etapa P-2: Procesamiento de la información mediante redundancia (§ 3.3.1).

6.1. Modelo de evaluación

¿Cómo calculamos las *disponibilidad de un objeto de información* en un sistema de almacenamiento? Como se demuestra en diversos estudios [9, 75, 23], la disponibilidad de la información se puede calcular a partir de un *modelo probabilístico*, del cual se estima la probabilidad P de que haya suficientes objetos de información disponibles en el sistema, para que cuando se solicite el objeto original éste se pueda reconstruir. Las variables de ese modelo probabilístico se describen en la Tabla 6.1.

Tabla 6.1: *Variables usadas por el modelo probabilístico* [9].

Variable	Descripción
$\mathcal{N}$	Número de nodos presentes en el sistema.
a	Disponibilidad promedio de los nodos.
$\mathcal{F}$	Número de archivos en el sistema.
A	Disponibilidad de la información.
k	Factor de redundancia, para estrategias sin códigos; factor de estiramiento, para estrategias basadas en códigos.
m	Número de unidades por objeto de información necesarias para su recuperación.
Y	Variable aleatoria para determinar el número de nodos disponibles de entre x seleccionados al azar.
Z	Variable aleatoria para la cantidad almacenamiento de información por nodo.

Supongamos que contamos con un sistema P2P en el que participan $\mathcal{N}$ nodos, sobre el cual distribuimos aleatoriamente una cantidad y de objetos de información a lo largo de y nodos del sistema. Si seleccionamos aleatoriamente x nodos, entonces la probabilidad de que exactamente y nodos de x estén disponibles es:

$$P(Y = y) = \binom{x}{y} a^y (1-a)^{(x-y)} \tag{6.1}$$

Donde $y < x \leq \mathcal{N}$. La Ec. (6.1) servirá para derivar fórmulas para la disponibilidad de la información, así como para la cantidad de almacenamiento y ancho de banda utilizados por cada estrategia de redundancia estudiada.

En lo que sigue calcularemos los factores de redundancia de información k, obtenidos a partir de la Ec. (6.1), para diferentes valores de disponibilidad de la información A, disponibilidad de los nodos a, entre otros parámetros que son particulares de cada una de las estrategias que contemplamos para la evaluación. Al final haremos una comparación con todos los resultados obtenidos.

6.2. Estrategias sin códigos

6.2.1. Redundancia simple

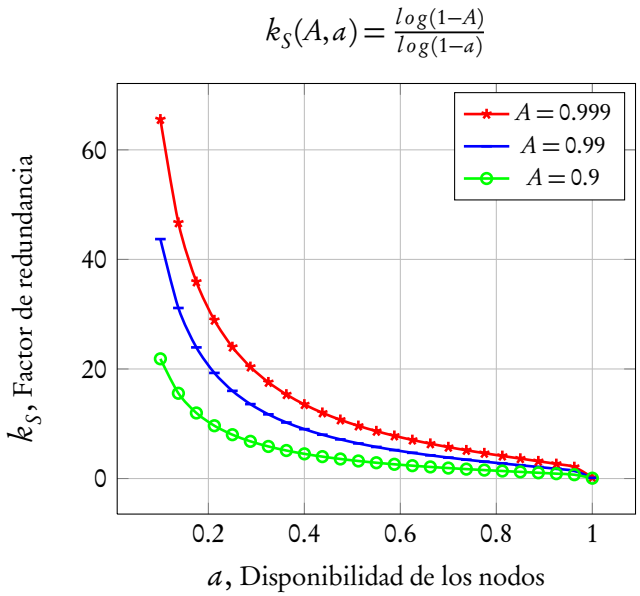
En § 3.4.1 describimos la aplicación de una estrategia de redundancia simple en un sistema de almacenamiento distribuido. De la Ec. (6.1) y considerando que si distribuimos k_S copias de un objeto de información, una copia para cada k_S nodos entre $\mathcal{N}$ posibles, la disponibilidad A del objeto se define como la probabilidad de que *uno o más nodos* entre los k_S nodos con una copia estén disponibles; esto es:

$$A = P(Y \geq 1) = 1 - (1-a)^{k_S} \tag{6.2}$$

Despejando el factor de redundancia k_S , tenemos que:

$$k_S(A,a) = \frac{log(1-A)}{log(1-a)} \tag{6.3}$$

Figura 6.1: *La gráfica muestra el efecto de variar el valor de A para* $k_S(A,a)$.



La Fig. 6.1 muestra el resultado de graficar el factor de redundancia $k_S(A,a)$ vs. la disponibilidad de los nodos a, para tres diferentes valores de disponibilidad de la información A. Allí se puede observar que con nodos más inestables, se necesita más redundancia para mantener niveles de disponibilidad de información cada vez más altos. Asimismo, con mayor estabilidad en los nodos, la redundancia decrece para los tres casos de A.

6.2.2. Redundancia con bloques

Figura 6.2: *La gráfica muestra el efecto de variar el valor de A para el mismo* $m = 10$*, en* $k_B(A, a, m)$.

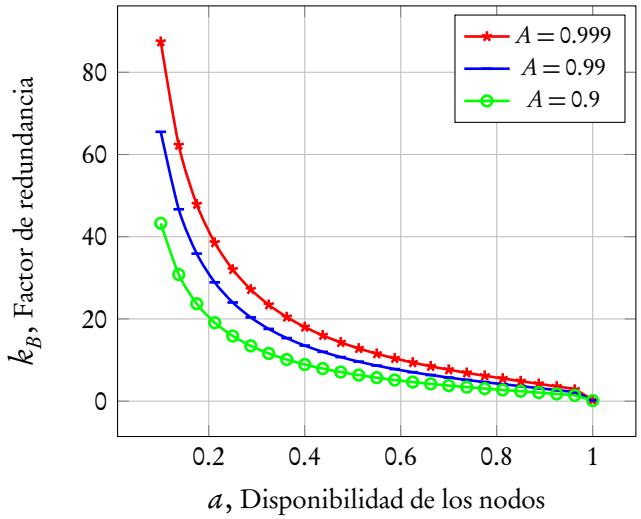
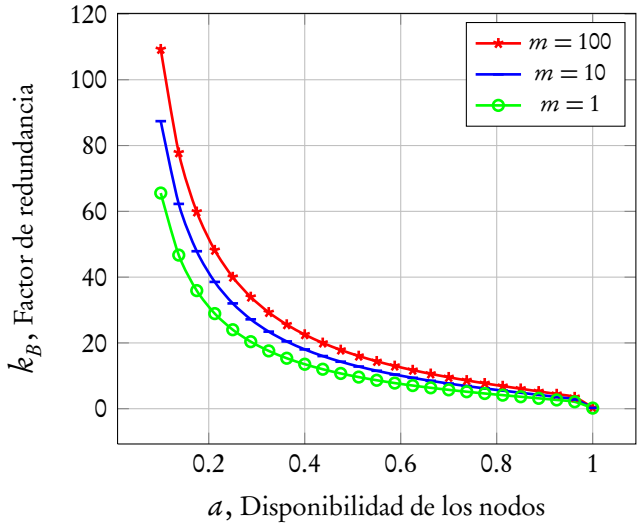


Figura 6.3: *La gráfica muestra el efecto de variar el valor de m para el mismo* $A = 0.999$*, en* $k_B(A, a, m)$.



En § 3.4.1 describimos la aplicación de una estrategia de redundancia por bloques en un sistema de almacenamiento distribuido. De la Ec. (6.1) y considerando que si distribuimos $m{\cdot}k_B$ copias de las unidades de un objeto de información, una para cada $m{\cdot}k_B$ nodos entre $\mathcal{N}$ posibles, la disponibilidad A del objeto se define como la probabilidad de que uno o más nodos entre los $m{\cdot}k_B$ nodos con una copia estén disponibles, para cada una de las m copias distribuidas por unidad; esto es:

$$A = P(Y \geq 1) = (1 - (1 - a)^{k_B})^m \tag{6.4}$$

Despejando el factor de redundancia k_B , tenemos que:

$$k_B(A,a,m) = \frac{log(1-A^{1/m})}{log(1-a)} \tag{6.5}$$

La Fig. 6.2 muestra la redundancia $k_B(A,a,m)$ vs. la disponibilidad de los nodos a, para tres diferentes valores de A. También se muestra que la inestabilidad de los nodos afecta la redundancia que se debe agregar al sistema para ofrecer la disponibilidad de información requerida. Mayor inestabilidad de los nodos implica mayor redundancia de información.

La redundancia con bloques también está influenciada por la cantidad de unidades m en las que se divide un objeto de información antes de distribuirse en el sistema. La Fig. 6.3 muestra el efecto de variar el valor de m utilizando la misma $A = 0.999$. Podemos notar que cuando $m = 1$ entonces $k_B = k_S$, y que con menor disponibilidad de los nodos, tener más unidades por objeto de información implica agregar mayor redundancia para conservar la misma disponibilidad de información.

6.3. Estrategias con códigos

6.3.1. Redundancia con IDA

En la Fig. 3.14 describimos como aplicar redundancia con IDA en un sistema de almacenamiento distribuido. De la Ec. (6.1) y considerando que si distribuimos $m \cdot k_I$ unidades codificadas —dispersos—, uno para cada $m \cdot k_I$ nodos entre $\mathcal{N}$ posibles, la disponibilidad A del objeto se define como la probabilidad de que por lo menos m de $m \cdot k_I$ nodos que almacenen un disperso estén disponibles; esto es:

$$A_I = P(Y \geq m) = \sum_{i=m}^{mk_I} \binom{mk_I}{i} a^i (1-a)^{mk_I - i} \tag{6.6}$$

Despejando el factor de estiramiento k_I , y suponiendo que $m \cdot k_I$ sea suficientemente grande para hacer una aproximación normal a esa distribución binomial, como se demuestra en [9], tenemos que:

$$k_I(A,a,m) = \left(\frac{\sigma(A)\sqrt{\frac{a(1-a)}{m}} + \sqrt{\frac{\sigma(A)^2 a(1-a)}{m} + 4a}}{2a} \right)^2 \tag{6.7}$$

Donde $\sigma(A)$ es la cantidad de desviaciones estándar en una distribución de probabilidad normal dado un nivel de disponibilidad requerido, y equivale a:

$$\sigma(A) = \sqrt{2}\,\mathrm{erf}^{-1}(-1+2A) \tag{6.8}$$

Las Figs. 6.4 y 6.5 muestran el factor de estiramiento $k_I(A,a,m)$ en función de la disponibilidad de los nodos a. En la primera variamos la disponibilidad de información A requerida, y en la segunda el número de unidades codificadas m necesarias para reconstruir el objeto original. Podemos deducir de esas gráficas que, al igual que en las estrategias de redundancia anteriores, la inestabilidad de los nodos hace que para mantener la disponibilidad de información deseada se deban aumentar los niveles de redundancia de información, sobre todo si A y m crecen. (Fig. 6.4)

Cabe mencionar que el caso cuando $m = 1$, en la Fig. 6.5, sólo tiene sentido en términos teóricos, no prácticos, en el caso de redundar información con IDA. Los límites teóricos del modelo probabilístico para k_I, ocurren cuando $m \to \infty$, y $a = 1$, esto es:

$$\lim_{m\to\infty} k_I(A,a,m) = \frac{1}{a},$$

Figura 6.4: *La gráfica muestra el efecto de variar el valor de A para el mismo* $m = 10$.

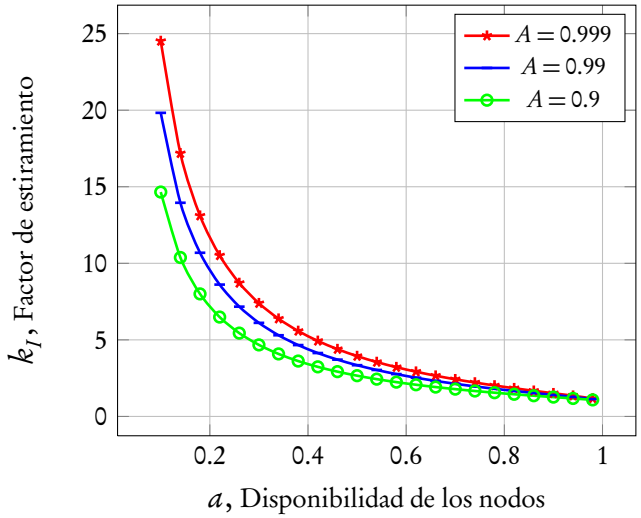
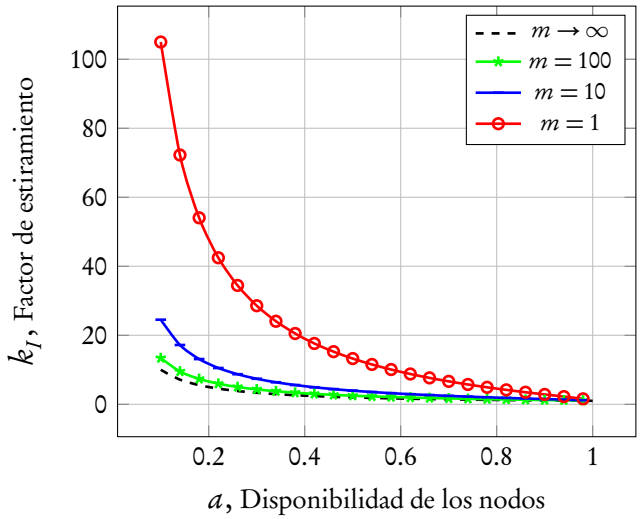


Figura 6.5: *La gráfica muestra el efecto de variar el valor de m para el mismo* $A = 0.999$.



y

$$k_I(A, 1, m) = 1.$$

En el primer caso se establece una cota teórica inferior para la redundancia en función del número de unidades necesarias para reconstruir el objeto original, como puede observarse en la Fig. 6.5 con la curva punteada equivalente a $1/a$. En el segundo, queda claro que cuando los nodos siempre están disponibles —no existen fallas— no se requiere redundancia extra.

6.3.2. Redundancia con fuente digital

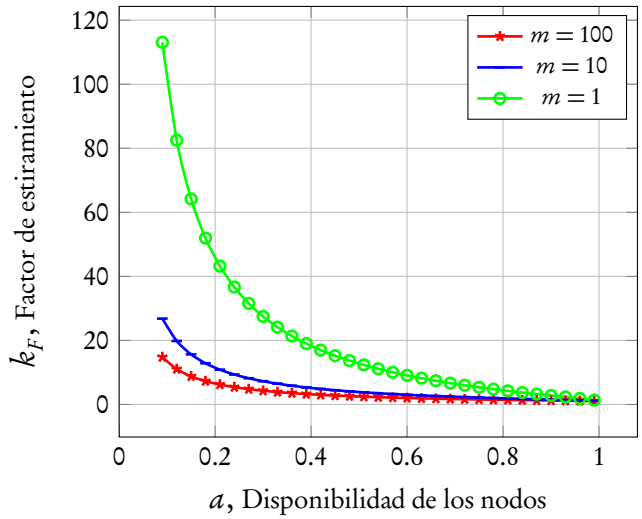
Las fuentes digitales son también códigos de borrado que, aunque no son tipo MDS como los códigos IDA, en términos de disponibilidad de la información, se definen de manera muy similar. En particular, con las

fuentes digitales es necesario reunir $m+\epsilon$ unidades codificadas para reconstruir el objeto original (§ 2.3.2); donde ϵ es el número de unidades extras para la recuperación.

Es difícil conocer de antemano el valor de ϵ. Sin embargo, es posible acotar su valor. En la práctica [49], el valor de ϵ suele ser 5 % del valor de m. Aunque esto depende del código usado para la fuente digital —e.g. Raptor, LT—, y del tamaño de m. Algunos estudios [90] indican que $m \simeq 10000$ para usar $\epsilon = 0.05m$. El factor de estiramiento de una fuente digital k_F se puede definir igual que k_I, pero considerando lo dicho arriba también está en función de ϵ; esto es:

$$k_F(A,a,m,\epsilon) = \left(\frac{\sigma(A)\sqrt{\frac{a(1-a)}{m(1+\epsilon)}} + \sqrt{\frac{\sigma(A)^2 a(1-a)}{m(1+\epsilon)} + 4a}}{2a} \right)^2 \tag{6.9}$$

Figura 6.6: *La gráfica muestra el efecto de variar el valor de m para el mismo* $A = 0.999$ *y* $\epsilon = 0.05$.



De la Fig. 6.6, al considerar la variable ϵ para calcular k_F, observamos diferencias marginales con k_I en la Fig. 6.5. En este sentido, la Fig. 6.7 muestra qué tan importante es el efecto de ϵ en la redundancia. Allí se puede apreciar que incluso cuando se requiere de un exceso de 50 % en la cantidad de paquetes para decodificar el objeto original, no hay una diferencia significativa con respecto a un exceso del 1 %, para $m = 10$. Además, sabemos por lo visto anteriormente, que si $m \to \infty$, $k_F = 1/a$, sin importar los valores de ϵ; en la Fig. 6.7 hay una curva para $m = 10000$ que se acerca a esa cota inferior.

6.3.3. Redundancia con códigos de red

Los códigos de red también son códigos de borrado, por lo que su redundancia también está definida como en la Ec. (6.7). Lo que diferencia el factor de estiramiento k_R para los códigos de red con respecto a k_I y k_F radica en que el número de unidades necesarias para recuperar la información original está en función, no sólo de m, sino también del orden p del campo con el que fue creado el código y del tamaño del objeto original L en bits.

En § 2.3.2 vimos que el objeto de información tiene un tamaño de N símbolos $s_{i,j} \in \mathbb{F}_p$, por lo que si $p = 2^b$, entonces $|s_{i,j}| = b$ bits. Ahora bien, si el objeto de información mide L bits en total, tenemos que $N = L/b$ símbolos. Sabiendo que para los códigos de red necesitamos reunir por lo menos $N/m = L/(b\,m)$

Figura 6.7: *La gráfica muestra el efecto de variar el valor de ϵ para el mismo $A = 0.999$ y $m = 10$.*

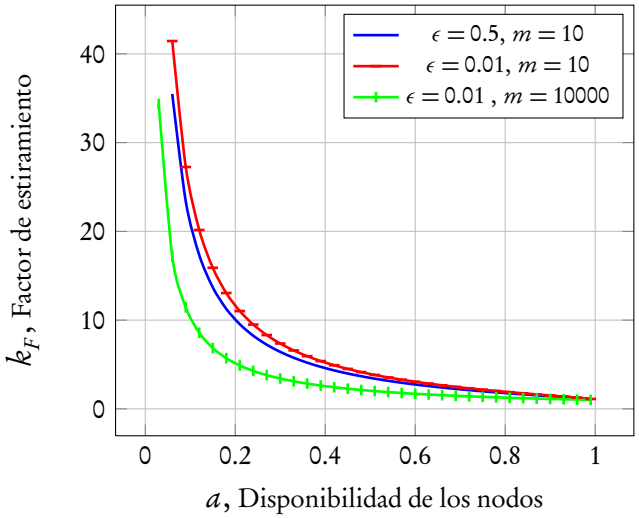
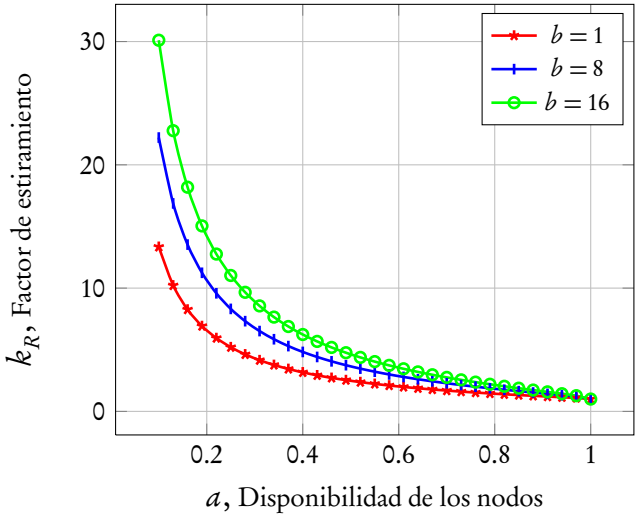


Figura 6.8: *Para el caso de esta gráfica, variamos el tamaño en bits por símbolo $s_{i,j}$.*



de $k_R N/m$ unidades codificadas para reconstruir el objeto original, y de acuerdo con la Ec. 6.7, podemos definir el factor de estiramiento para los códigos de red como

$$k_R(A,a,m,L,b) = \left(\frac{\sigma(A)\sqrt{\frac{a(1-a)}{L/(bm)}} + \sqrt{\frac{\sigma(A)^2 a(1-a)}{L/(bm)} + 4a}}{2a} \right)^2 \tag{6.10}$$

Podemos ver de la Ec. (6.10) que si $L \to \infty$ entonces $k_R = 1/a$. Esta vez, el factor mb representa el tamaño en bits de cada unidad de codificada mediante códigos de red (Fig. 3.16). En particular, para un tamaño fijo de $m = 100$ símbolos por unidad, $L = 10240$ bits totales por objeto, y para una disponibilidad de información requerida de $A = 0.999$, vemos en la Fig. 6.8, que al aumentar el tamaño b de cada símbolo $s_{i,j}$, aumenta el valor del factor k_R.

6.4. Redundancia híbrida

La redundancia híbrida suma la redundancia generada por una estrategia IDA más la generada por una estrategia simple cuando $k_S = 1$. De este modo,

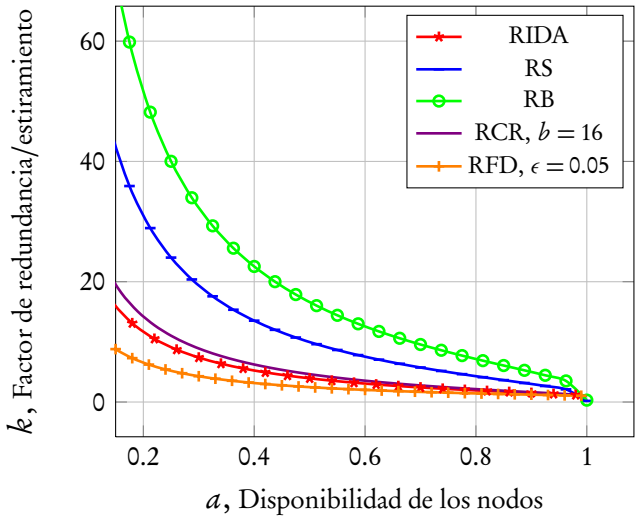
$$k_H(A,a,m) = k_I(A,a,m) + 1 \tag{6.11}$$

Consideramos que no vale la pena presentar gráficas de k_H porque las de k_I son igual de representativas de los efectos de variar A y m en función de la disponibilidad de los nodos.

6.5. Comparación de estrategias

La Fig. 6.9 muestra los factores de redundancia y estiramiento para las estrategias que hemos evaluado bajo los siguientes parámetros: $A = 0.999$, $m = 100$. Por las razones expuestas anteriormente, no incluimos curvas para k_H. Queda claro, luego de observar esa gráfica, que las estrategias que no utilizan códigos, sino copias, para redundar la información, exigen que se agregue más redundancia al sistema en la medida que la disponibilidad de los nodos decrece, con respecto a las que sí utilizan códigos.

Figura 6.9: *Comparación entre diferentes factores de redundancia, k_S, k_B, k_I, k_R, k_F.*



Por su parte, las estrategias que utilizan códigos son superiores aún en condiciones tan difíciles como cuando sólo el 20 % de los nodos está disponible. No obstante, cuando los nodos son más estables, todas las estrategias presentan condiciones similares de redundancia; después que la disponibilidad de los nodos comienza a ser mayor del 80 %, no hay más de 10 unidades de diferencia entre los factores de redundancia, y estos disminuyen aún más cuando a se acerca a 1.

También puede observarse que las estrategias basadas en códigos no difieren en gran medida entre sí. En la Fig. 6.9 mostramos k_F con $\epsilon = 0.05$; si $m = 100$, significa que se requeriran al menos 105 unidades codificadas para reconstruir el objeto original. Para otros valores de ϵ no hay diferencias significativas, como de mostró en la Fig. 6.6, sin embargo, las fuentes digitales, considerando su factor k_F, pueden considerarse más aptas que las demás estrategias bajo condiciones de red difíciles y con las mismas exigencias de disponibilidad de la información. En cuanto a los códigos de red, k_R, con $b = 16$ bits por símbolo, difiere relativamente poco con respecto a k_I, y las diferencias son incluso menores para otros valores de b.

Aquí cabe recalcar que todos los resultados mostrados en este capítulo están basados el modelo probabilístico descrito en [9]. Y que agregamos resultados y comparaciones a partir de dicho modelo que incluyen redundancia con fuentes digitales y códigos de red que no se encuentran detallados en la literatura. En particular, consideramos como aportación de este trabajo de investigación, las Figs. 6.6, 6.7, 6.8 y 6.9. Así como la particularización de la Ec (6.7) para fuentes digitales y códigos de red en las Ecs. (6.9) y (6.10), respectivamente.

Capítulo 7

Evaluación del almacenamiento

En este capítulo evaluamos la Etapa D-2: Envío y almacenamiento de la información (§ 3.3.2).

7.1. Modelo de evaluación

En este capítulo estudiaremos la cantidad de almacenamiento de información generado por utilizar alguna de las estrategias de redundancia vistas anteriormente. En particular, sólo mostraremos las estrategias de redundancia simple, con IDA e híbrida, porque son las que nos ofrecen resultados significativos en términos de la manera en la que se almacena la información.

Podemos usar diversas unidades de medición de almacenamiento, e.g., bytes, símbolos, número de objetos de información. También podemos considerar el almacenamiento total en el sistema o el promedio por nodo. Para efectos de este trabajo de investigación, consideraremos el almacenamiento promedio por nodo medido en bytes.

Teóricamente, como se indica en [9], con la política de asignación aleatoria de y objetos de información sobre exactamente y nodos, es posible obtener un límite en la cantidad de objetos que un nodo almacena. Así la probabilidad de que un nodo almacene exactamente z, o a lo más z objetos, está dada por las Ecs. (7.1) y (7.2), respectivamente.

$$P(Z = z) = \frac{(y/\mathcal{N})^z}{z!} e^{-y/\mathcal{N}} \tag{7.1}$$

$$P(Z \leq z) = \sum_{i=0}^{z} \frac{(y/\mathcal{N})^i}{i!} e^{-y/\mathcal{N}} \tag{7.2}$$

Estas ecuaciones siguen una distribución de probabilidad de Poisson, con media $y/\mathcal{N}$ y varianza $y/\mathcal{N}$.

Si un nodo almacena d bytes, ¿cuántos bytes $\overline{d}$ almacena en promedio un nodo de un sistema P2P que sigue el modelo probabilístico descrito en § 5? Si k es el factor de redundancia o estiramiento de la estrategia de redundancia usada, entonces

$$\overline{d} = k \times \frac{\text{Total de objetos de información en el sistema} \times \text{Tamaño del objeto}}{\text{Número de nodos de almacenamiento en el sistema}} \tag{7.3}$$

Siendo k el único factor en la Ec. (7.3) que hace la diferencia en el impacto del almacenamiento de información entre las estrategias de redundancia, y en conformidad con la Fig. 6.9, evaluaremos RIDA y RS porque son las estrategias que ofrecen resultados en redundancia representativos por el uso, o no, de códigos. Sin embargo, en cuanto a usar redundancia híbrida para almacenamiento, $\overline{d}$ difiere del que se define en

la Ec. (7.3),

$$\overline{d}_H = \overline{d}_I + \frac{\text{Total de objetos de información en el sistema} \times \text{Tamaño del objeto}}{\text{Número de nodos de almacenamiento en el sistema}} \tag{7.4}$$

Donde $\overline{d}_I$ es el almacenamiento promedio por nodo cuando usamos la estrategia IDA. En la Tabla 7.1 definimos el almacenamiento promedio por nodo de cada una de las estrategias que evaluaremos.

Tabla 7.1: *Almacenamiento promedio por nodo.*

Estrategia	$\overline{d}$ [bytes]
RS	$\overline{d}_S = k_S(A,a)\frac{\mathcal{F}M}{\mathcal{N}}$
RIDA	$\overline{d}_I = k_I(A,a,m)\frac{\mathcal{F}M}{\mathcal{N}}$
RH	$\overline{d}_H = [k_I(A,a,m)+1]\frac{\mathcal{F}M}{\mathcal{N}}$

Al igual que en el trabajo de [9], realizaremos una simulación de tipo Monte-Carlo para evaluar el almacenamiento promedio por nodo. Nuestro escenario de simulación presenta las características descritas en la Tabla 7.2. El tamaño de los objetos de información sigue una distribución de probabilidad normal, mientras que la selección y disponibilidad de los nodos en los que se almacenan los objetos están determinadas por una distribución de probabilidad uniforme.

Tabla 7.2: *Variables usadas para la simulación de almacenamiento P2P.*

Variable	**Valor**
$\mathcal{N}$	1000 nodos.
$\mathcal{F}$	1000 objetos de información.
M	700 MB por objeto.
σ_M	$\pm$100 MB por objeto.
A	0.9 y 0.999 por estrategia.
a	0.3, 0.5 y 0.9 por estrategia.

Los resultados de cada simulación son presentandos a través de una figura en la que se incluye un histograma del almacenamiento por nodo, obtenido después de medir la cantidad de bytes que cada nodo almacenó al final de cada ejecución de la simulación. La figura con el histograma también incluye un valor para el almacenamiento promedio por nodo $\overline{d} \pm \varsigma$, caracterizado por cierto *intervalo de confianza*.

Un intervalo de confianza especifica los límites entre los cuales un valor estimado tiene las mayores probabilidades de estar ubicado. En nuestro caso, dicho valor es el almacenamiento promedio por nodo $\overline{d}$, y el intervalo de confianza que obtenemos para él está calculado siguiendo el método descrito en [77]:

- Escogemos un valor aceptable ς para la desviación estándar de nuestro valor estimado $\overline{d}$.
- Generamos al menos 30 valores de $\overline{d}$ mediante las simulaciones.
- Continuamos generando valores de $\overline{d}$ hasta tener j de ellos de modo que $c\mathcal{S}/\sqrt{j} \leq \varsigma$, donde $c > 0$ es una constante que equivale a 1.96 cuando requerimos una confianza del 95 % para nuestro valor

estimado; además

$$S^2 = \frac{\sum_{i=1}^{j}(x_i - \overline{x})^2}{j-1}, \tag{7.5}$$

S es la *desviación estándar muestral*, y

$$\overline{x} = \sum_{i=1}^{j} \frac{x_i}{j}, \tag{7.6}$$

es la *media muestral* que estima el valor $\overline{d}$.

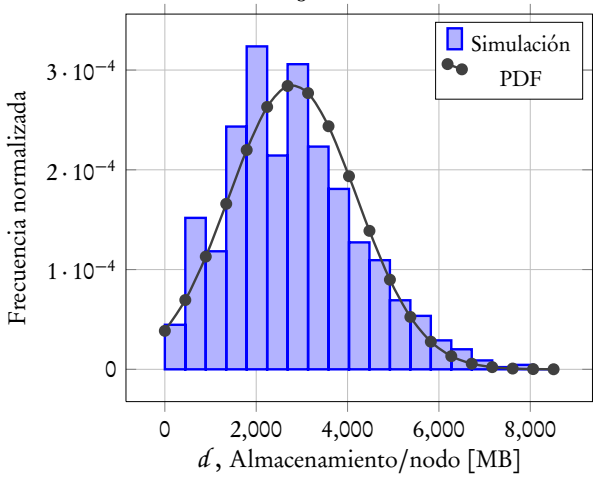
- Cuando la restricción $cS/\sqrt{j} \leq \varsigma$ se vea cumplida, entonces decimos que el valor $\overline{d}$ está comprendido en el intervalo

$$\overline{x} - 1.96S/\sqrt{j} \leq \overline{d} \leq \overline{x} + 1.96S/\sqrt{j}$$

con una probabilidad del 95 %.

7.2. Redundancia simple

Figura 7.1: $A = 0.9$, $a = 0.5$, $k_S = 5$, $\overline{d} = 2800.12 \pm 4.88MB$



Las Figs. 7.1 y 7.2 muestran los resultados de la simulación de un sistema de almacenamiento P2P con los características de la Tabla 7.2. En ellas puede observarse que el almacenamiento promedio por nodo $\overline{d}$ cambia al aumentar los requerimientos de disponibilidad de la información. Con $A = 0.9$, $\overline{d} = 2800.12 \pm 4.88MB$; con $A = 0.9999$, $\overline{d} = 9811.29 \pm 11.91\,MB$. Como se esperaba, debido al factor de redundancia k_S, se requiere almacenar más información en los nodos para soportar una mayor disponibilidad de información.

Ahora bien, ¿qué pasa con $\overline{d}$ cuando la disponibilidad de los nodos cambia, pero la disponibilidad de la información se mantiene constante? Las Figs. 7.3 y 7.4, muestran que para soportar la disponibilidad de información deseada, $A = 0.9999$, el almacenamiento aumenta considerablemente cuando los nodos de almacenamiento se vuelven inestables porque ofrecen poca disponibilidad; el efecto inverso ocurre cuando la disponibilidad de los nodos aumenta para conformar un sistema P2P más estable.

Las curvas que acompañan los histogramas representan las funciones de densidad de probabilidad (PDF) normal con media y desviación estándar son las de la aproximación normal a la distribución de Poisson de las Ecs. (7.1) y (7.2). De esta manera, que curvas —resultados teóricos— e histogramas —resultados

Figura 7.2: $A = 0.9999$, $a = 0.5$, $k_S = 14$, $\overline{d} = 9811.29 \pm 11.91\ MB$

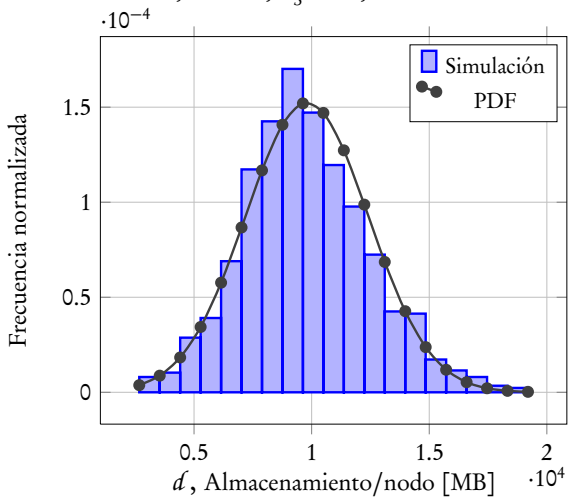


Figura 7.3: $A = 0.9999$, $a = 0.3$, $k_S =$, $\overline{d} = 18193.67 \pm 18.56\ MB$

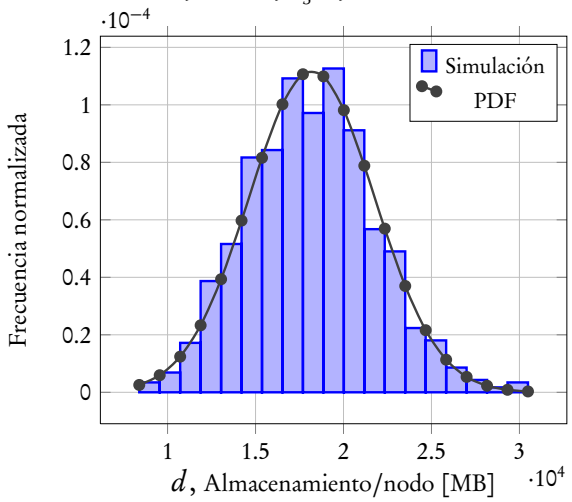


Figura 7.4: $A = 0.9999$, $a = 0.9$, $k_S = 14$, $\overline{d} = 3500.78 \pm 4.02\ MB$

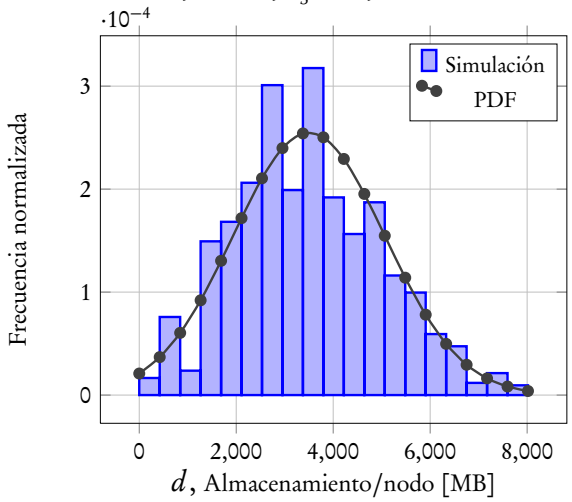
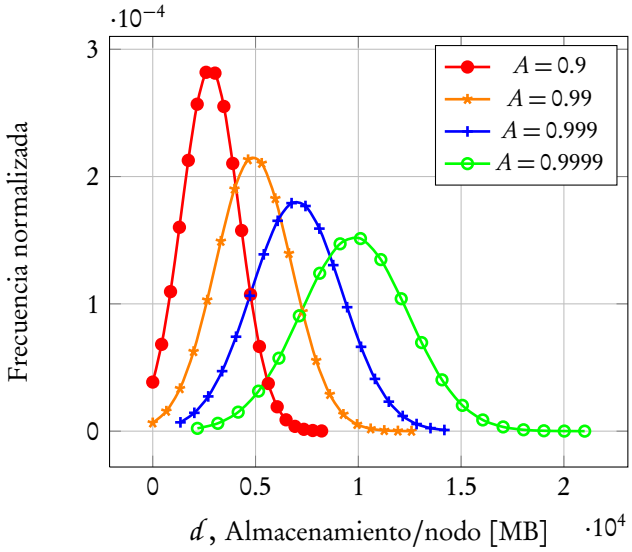


Figura 7.5: *Comparación de las PDF para distintos requerimientos de disponibilidad de la información con RS.*



experimentales— se aproximen, significa que el resultado de las simulaciones valida los aspectos teóricos sobre redundancia y disponibilidad de la información planteados hasta ahora. La Fig. 7.5 muestra una familia de curvas de PDF de almacenamiento para RS.

7.3. Redundancia por IDA

Bajo los mismos escenarios considerados en la sección anterior, presentamos las Figs 7.6 y 7.7, donde se muestra que las mayores exigencias en disponibilidad de la información no impactan de manera tan importante como lo hacen con la redundancia simple. Si variamos la disponibilidad de los nodos —$a = 0.3$ y $a = 0.9$, en las Figs 7.8 y 7.9, respectivamente— notamos que $\overline{d}$ disminuye tanto que casi se acerca al tamaño original de los archivos cuando la disponibilidad de los nodos es muy alta.

Figura 7.6: $A = 0.9$, $a = 0.5$, $k_I = 2.19$, $m = 100$, $\overline{d} = 1531.50 \pm 2.70\ MB$

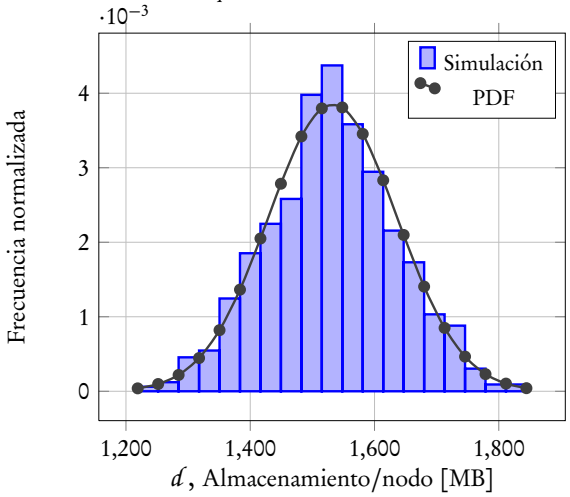


Figura 7.7: $A = 0.9999$, $a = 0.5$, $k_I = 2.60$, $m = 100$, $\overline{d} = 1818.91 \pm 3.25\ MB$

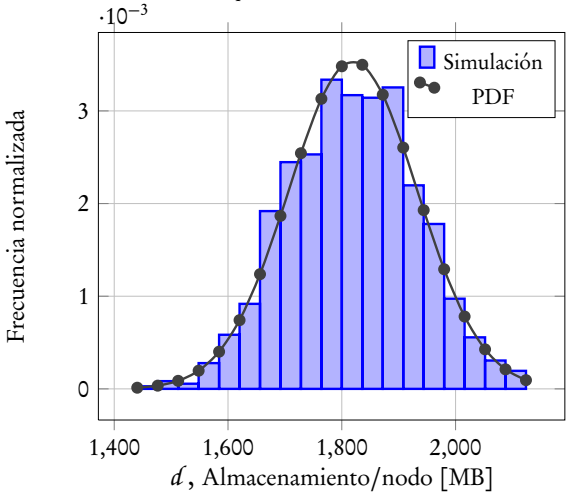


Figura 7.8: $A = 0.9999$, $a = 0.3$, $k_I = 4.54$, $m = 100$, $\overline{d} = 3183.48 \pm 4.24\ MB$

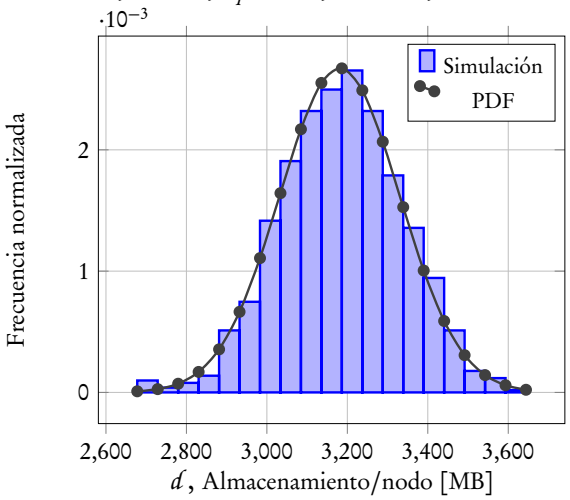


Figura 7.9: $A = 0.9999$, $a = 0.9$, $k_I = 1.25$, $m = 100$, $\overline{d} = 873.04 \pm 1.29\ MB$

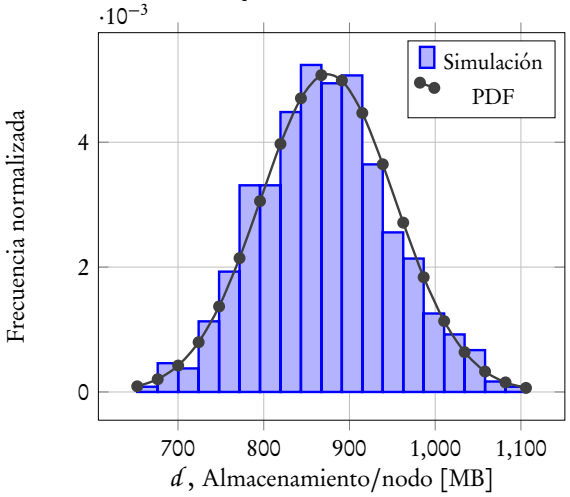
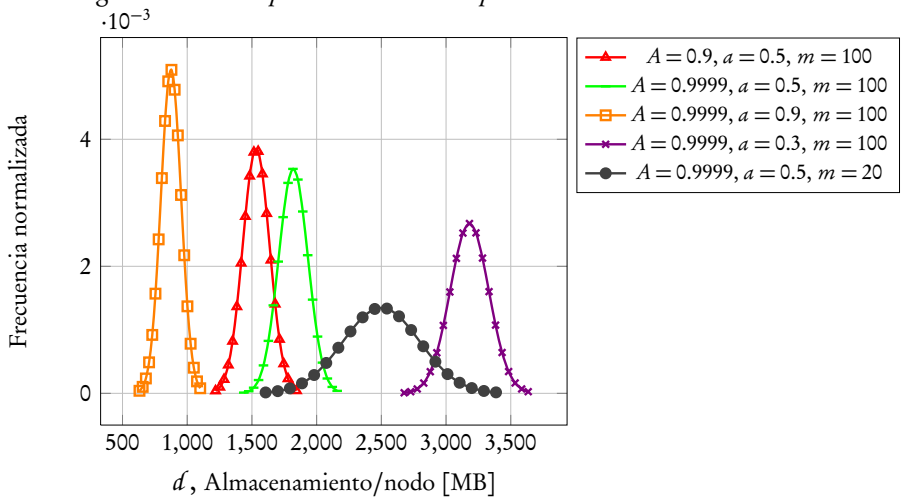


Figura 7.10: *Comparación entre PDF para almacenamiento con RIDA.*

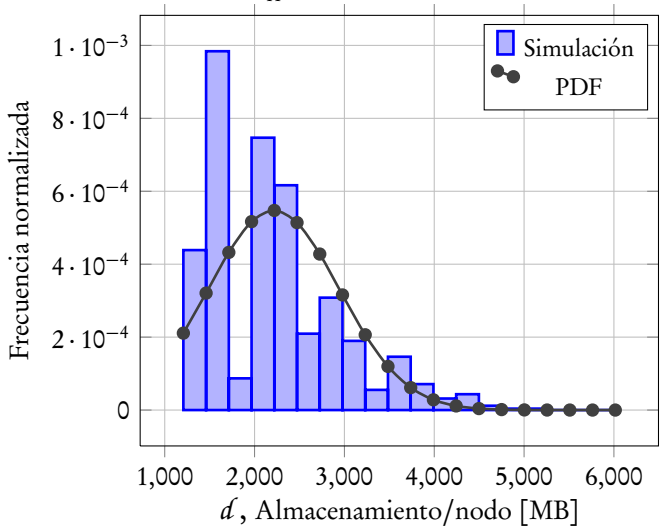


La Fig. 7.10 muestra una comparativa entre diferentes curvas de funciones de densidad de probabilidad. Allí también puede verse el efecto de mantener constante A y variar a, con las curvas claramente separadas entre sí, la combinación $A = 0.9999$, $a = 0.9$, $m = 100$ ofrece el $\overline{d}$ más bajo, mientras que la inestabilidad de los nodos con la combinación de parámetros $A = 0.9999$, $a = 0.3$, $m = 100$ hace crecer enormente los requerimientos de almacenamiento. Asimismo, cuando disminuye la cantidad de unidades m necesarias para recuperar la información original, $\overline{d}$ con $m = 20$ también aumenta con respecto a $m = 100$.

7.4. Redundancia híbrida

En lo que sigue presentamos una serie de figuras que resultan de la simulación del almacenamiento por redundancia híbrida bajo los mismos escenarios planteados para RIDA (Figs. 7.11, 7.12, 7.13 y 7.14).

Figura 7.11: $A = 0.9$, $a = 0.5$, $k_H = 3.19$, $m = 100$, $\overline{d} = 2233.95 \pm 2.24\ MB$



De la Fig. 7.15 vemos que las curvas de las diferentes PDF se encuentran a menor distancia entre sí con respecto a las que se muestran en la Fig. 7.10, aunque conservan el mismo orden de izquierda a derecha.

Figura 7.12: $A = 0.9999$, $a = 0.5$, $k_H = 3.60$, $m = 100$, $\overline{d} = 2524.65 \pm 4.47\ MB$

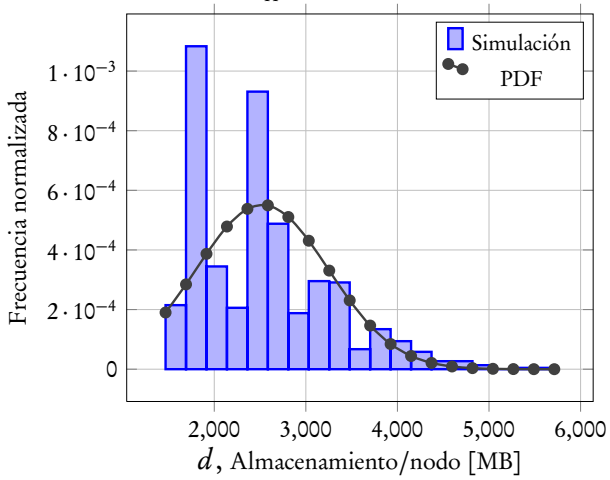


Figura 7.13: $A = 0.9999$, $a = 0.3$, $k_H = 5.54$, $m = 100$, $\overline{d} = 3883.34 \pm 5.25\ MB$

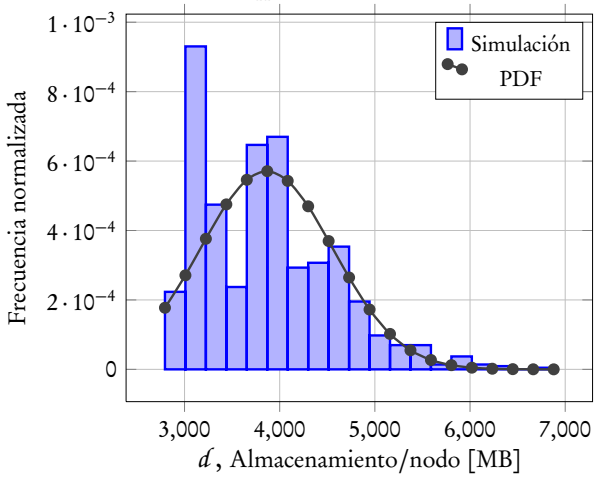


Figura 7.14: $A = 0.9999$, $a = 0.9$, $k_H = 2.25$, $m = 100$, $\overline{d} = 1574.28 \pm 2.35\ MB$

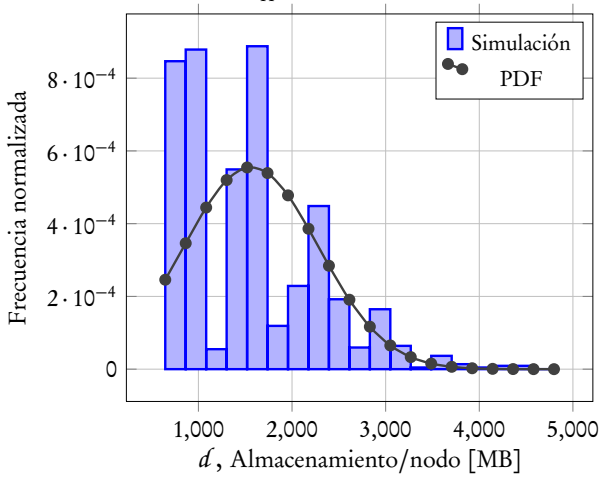
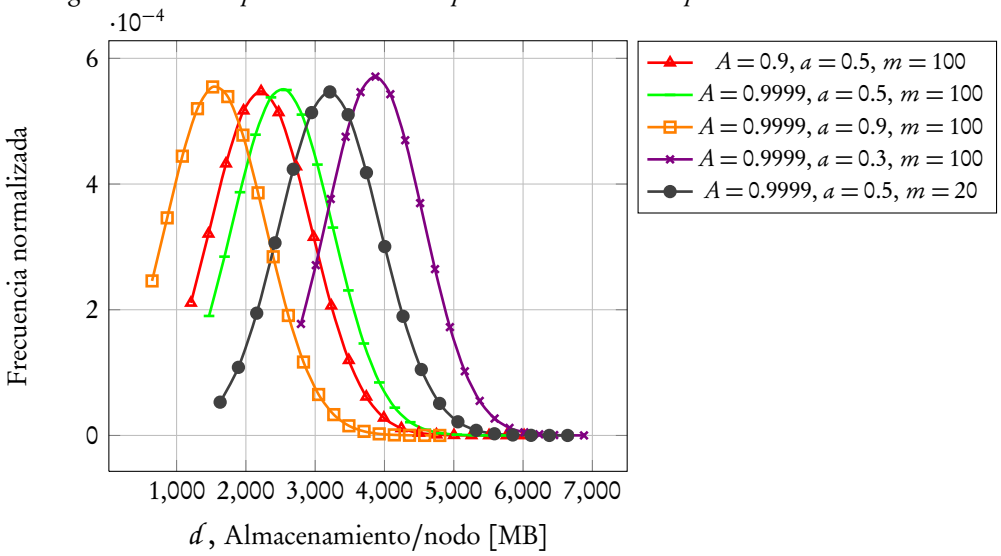


Figura 7.15: *Comparación entre PDF para almacenamiento por redundancia híbrida.*

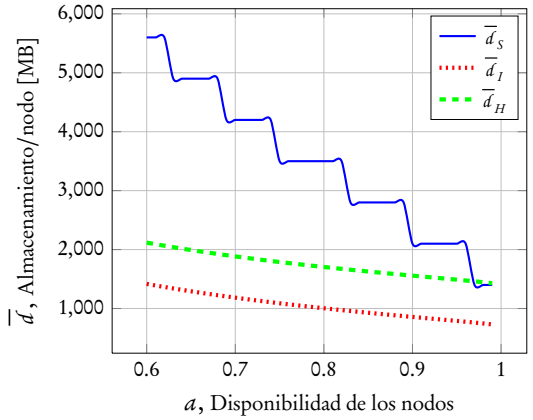


Lo anterior muestra que mantener una copia extra de cada archivo en el sistema afecta la distribución de la información almacenada. En particular, el almacenamiento promedio por nodo $\overline{d}$ con redundancia híbrida tiene un aumento constante con respecto a IDA, tal como se predice con $\overline{d}_H$ en la Tabla 7.1.

7.5. Comparación de estrategias

La Fig. 7.16 muestra una comparación entre las estrategias revisadas en las simulaciones, donde se grafica el almacenamiento promedio por nodo $\overline{d}$ en función de la disponibilidad de los nodos, siguiendo las fórmulas de la Tabla 7.1. Allí puede verse, como se esperaba, que no existe una diferencia importante entre RIDA y RH en términos de la cantidad de almacenamiento. En la medida que $a \to 1$, el almacenamiento promedio por nodo para RS, RIDA y RH, —$\overline{d}_S$, $\overline{d}_I$ y $\overline{d}_H$, respectivamente—, se acerca entre sí en todas las estrategias porque se requiere menor redundancia debido a la estabilidad de los nodos. Evidentemente, el almacenamiento promedio por nodo $\overline{d}$ es directamente proporcional al factor de redundancia de cada estrategia por un factor $\frac{\mathcal{F}M}{\mathcal{N}}$ (Tabla 7.1).

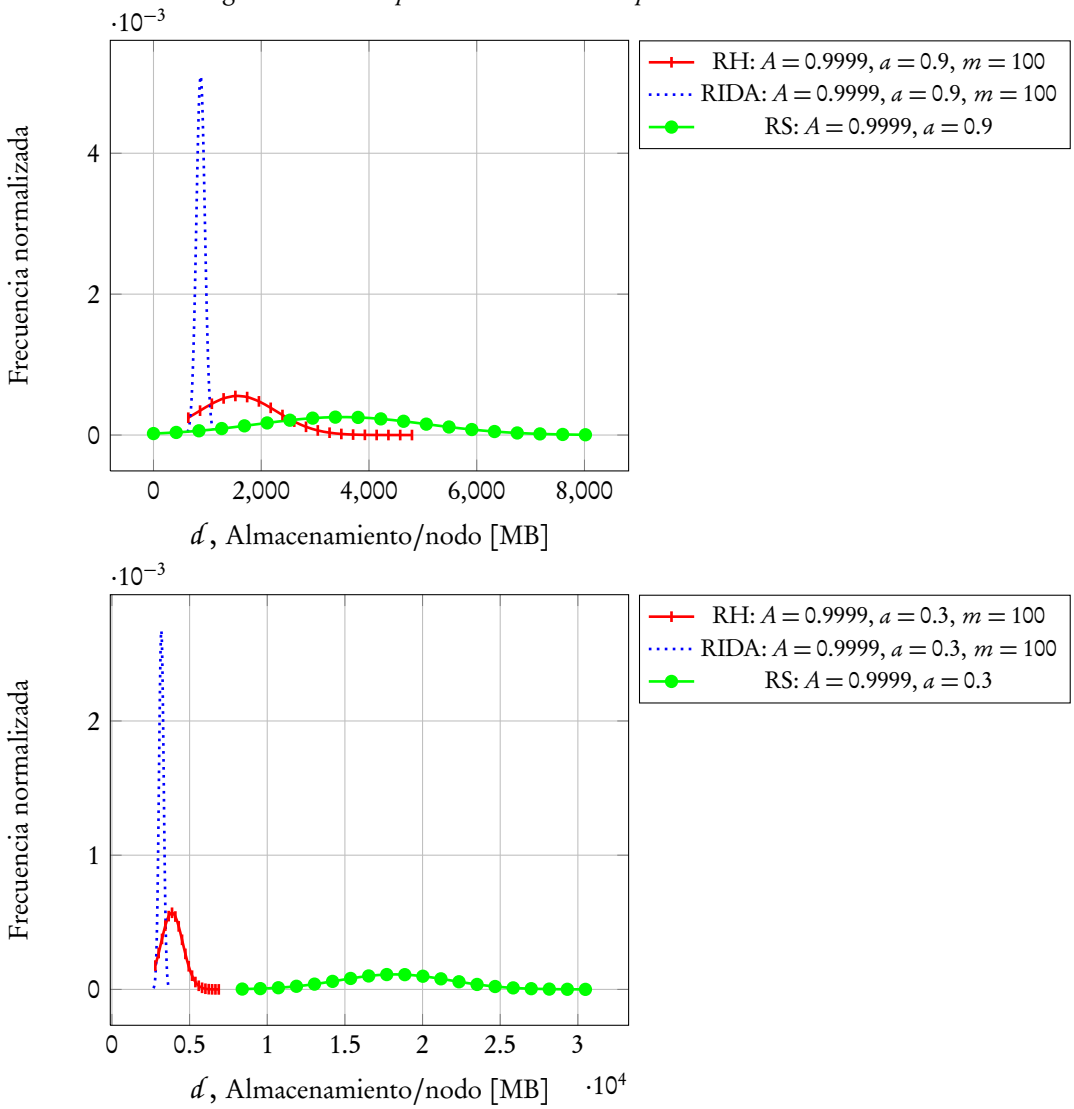
Figura 7.16: *Almacenamiento promedio por nodo, donde* $A = 0.999$, $m = 100$, $\mathcal{N} = 1000$ *y* $\mathcal{F} = 1000$.



En la Fig. 7.17 se hace una comparación de las tres estrategias de redundancia estudiadas para los casos en los que $a = 0.9$ y $a = 0.3$, arriba y abajo, respectivamente. Puede verse de allí, de izquierda a derecha, que las curvas indican que RIDA es la que menor cantidad de almacenamiento demanda para las mismas condiciones de disponibilidad de información, aún en condiciones extremas de disponibilidad de los nodos. Esta tendencia en el orden en el que aparecen las curvas se mantiene aún cuando cambiamos las disponibilidad de la información requerida; así que en la Fig. 7.17 sólo mostramos el efecto cuando $A = 0.9999$. Sin embargo, la estrategia híbrida requiere un mayor almacenamiento en relación con RIDA; entre 20 % y 80 % más para $a = 0.3$ y $a = 0.9$, respectivamente.

Las curvas también muestran, con la anchura de las campanas, que la información se distribuye de manera más balanceada cuando se utiliza RIDA. A diferencia de la estrategia simple, donde la campana que la caracteriza se extiende a lo largo de un amplio rango de valores de almacenamiento por nodo, con respecto a las otras estrategias. Lo anterior puede deberse al tamaño de los objetos de información que son distribuidos con una u otra estrategia. Con RIDA los objetos se dividen en m unidades de información con un tamaño promedio de $M/m \approx 7\ MB$, mientras que con redundancia simple, ese tamaño es $M \approx 700\ MB$.

Figura 7.17: *Comparación de entre PDF para almacenamiento.*



Aunque buena parte de los resultados presentados en este capítulo pueden deducirse a partir de lo dicho en § 6, consideramos importante ofrecer un capítulo enfocado en la evaluación del almacenamiento porque así logramos dimensionar con mayor profundidad y precisión los impactos de cada estrategia. En particular, consideramos como aportación de este trabajo de investigación cada una de las figuras mostradas —para el espectro de valores usados en la simulación—, especialmente las que tienen que ver con las comporaciones (Figs. 7.16 y 7.17), así como las estimaciones hechas con intervalos de confianza de los valores de almacenamiento promedio por nodo $\overline{d}$.

Capítulo 8

Evaluación del ancho de banda

En este capítulo evaluamos la Etapa M-3: Mantenimiento y restauración de la información (§ 3.3.3).

8.1. Modelo de evaluación

Cuando un nodo integrante de un sistema de almacenamiento distribuido deja de ofrecer la información que almacena —bajo ciertos criterios predefinidos—, entonces se dice que *falla*. Aunque las causas de esa falla pueden ser diversas, el efecto es el mismo: la *pérdida* de la información contenida en el nodo.

En § 3.1 explicamos que en los sistemas P2P esas fallas son una norma, más que una excepción. En § 3.3.3, por otra parte, expusimos que un sistema de almacenamiento debe proveer mecanismos para preservar la redundancia perdida por causa de las fallas en los elementos que lo conforman. Allí mismo ofrecimos una solución genérica para preservar la información, la cual está basada en tres etapas: monitoreo, elección y restauración de la información.

En este capítulo estudiaremos el impacto que hay en el uso del ancho de banda de cada una de las estrategias de redundancia vistas hasta ahora. Pero no incluimos resultados relacionados con la distribución de la información (Etapa D-2), sino los que tienen que ver con el *mantenimiento de la redundancia*, específicamente, con la restauración de la redundancia perdida (Etapa M-3). Lo anterior se debe a que nos interesa conocer las capacidades y comportamiento de cada estrategia de redundancia para enfrentar los embates de las fallas que caracterizan un sistema P2P.

¿Cómo modelar el comportamiento de un sistema de almacenamiento P2P de tal modo que podamos obtener resultados sobre los efectos del mantenimiento de la información en el ancho de banda? ¿Qué aspectos deben tomarse en cuenta para ese modelo? En § 5 revisamos brevemente una propuesta diseñada por [12] para dicho modelo, de la cual hacemos una descripción más detallada a continuación. El modelo de mantenimiento de información creado por [12] hace las siguientes asunciones:

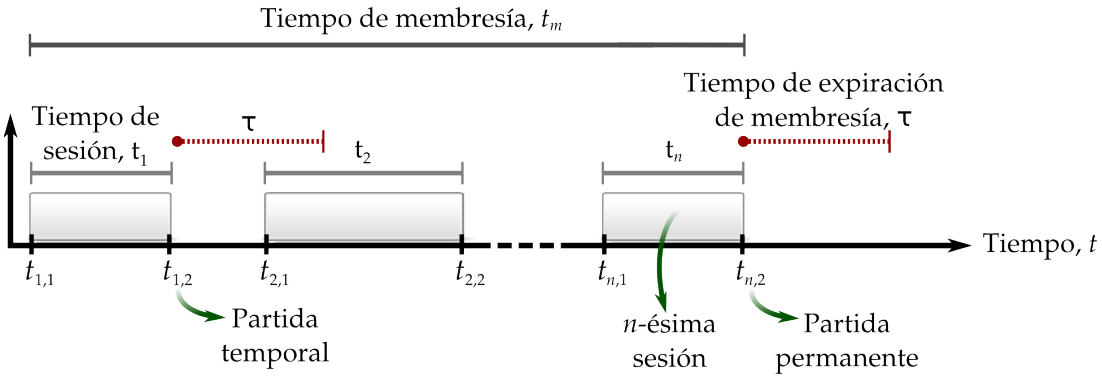
- Existe un sistema de almacenamiento P2P que consiste de una colección grande y dinámica de nodos que almacenan información cooperativamente.
- Los resultados obtenidos del modelo están centrados en los casos promedio. En particular, el almacenamiento y el ancho de banda promedio por nodo, $\overline{d}$ y $\overline{b}$, respectivamente.
- Hay una tasa constante de entradas y salidas de nodos en el sistema, λ y μ, respectivamente. Además, $\lambda = \mu$, por lo que hay un número constante de nodos en el sistema. En promedio —usando un resultado conocido de teoría de colas, ley de Little [44]— un nodo permanece como miembro del sistema por un periodo de tiempo $T = \mathcal{N}/\lambda$.
- El sistema almacena un total de $\mathcal{D}$ bytes con información única que fue distribuida con cierto factor de redundancia o estiramiento, k, para un total de $\mathcal{D}_t = k\mathcal{D}$ bytes de almacenamiento contribuido.

- Si un nodo se *une* al sistema, éste debe descargar para sí toda la información que ofrecerá posteriormente. En promedio, esa transferencia de información es de $\mathcal{D}_t/\mathcal{N}$. De este modo, se asume una contribución idéntica de almacenamiento por nodo.
- Cuando da por hecho que un nodo *parte* del sistema, toda la información que almacenó es entregada por el sistema a un nodo recién llegado. En este caso, se realiza otra transferencia de $\mathcal{D}_t/\mathcal{N}$ bytes.
- El ancho de banda utilizado para mantener la redundancia perdida, luego de la partida de algún nodo, es la suma de la cantidad de información transferida por unidad de tiempo causada por la entrada y salida de nodos. Dado que los nodos entran y salen cada $1/\lambda = 1/\mu$ unidades de tiempo, dicho ancho de banda es de $2\lambda\mathcal{D}_t/\mathcal{N} = 2\mathcal{D}_t/T$ bytes por unidad de tiempo. Dicho de otra manera, el ancho de banda promedio por nodo $\overline{b}$ para mantener la redundancia de información equivale a

$$\begin{aligned}\overline{b} &= 2 \times \frac{\text{Cantidad de información en el sistema}}{\text{Tiempo de membresía del nodo en el sistema}} \\ &= 2\frac{\mathcal{D}_t}{T} \\ &= 2\frac{k\mathcal{D}}{T}.\end{aligned} \tag{8.1}$$

Un aspecto clave para utilizar el modelo descrito arriba es la distinción entre *partidas temporales* y *partidas permanentes* de los nodos. Para ello, también es importante definir el *tiempo de membresía* de un nodo: «es el periodo de tiempo que inicia desde que un nodo se unió al sistema, y termina hasta el momento en el que parte para no volver». Bajo esta definición un nodo puede tener una o más sesiones en las cuales entra y sale temporalmente del sistema, hasta una última sesión, tal y como se muestra en la Fig. 8.1.

Figura 8.1: *Distinción entre partidas temporales y permanentes* [75].



En correspondencia con lo anterior, podemos definir la disponibilidad a de *un solo nodo* como la suma de los tiempos de sesión t_i de un nodo divididas por su tiempo de membresía t_m en el sistema. De acuerdo con lo que se muestra en la Fig. 8.1, esto es

$$a = \frac{\sum_{i=1}^{n} t_i}{t_m} = \frac{\sum_{i=1}^{n}(t_{i,2} - t_{i,1})}{t_m}. \tag{8.2}$$

Ahora bien, ¿cómo distinguir, *en la práctica*, la partida permanente de un nodo de la que es temporal? Esto se logra mediante una heurística: una vez que un nodo parte en el tiempo $t_{i,2}$, se inicia un temporizador

con duración τ; si el nodo no llega antes de $t_{i,2}+\tau$, entonces se considera que partió permanentemente. Con el tiempo de expiración de membresía τ, le Ec. 8.1 queda de la siguiente manera:

$$\bar{b}(\tau) = 2\frac{k(A,a(\tau))\,\mathcal{D}}{T(\tau)}. \tag{8.3}$$

Donde $\mathcal{N}(\tau)$ es el número de nodos en el sistema que no han agotado su membresía con respecto a τ. $T(\tau)$ es el periodo de tiempo que pasan los $\mathcal{N}(\tau)$ nodos en promedio en el sistema. Y la función $a(\tau)$ es la disponibilidad *promedio* de los $\mathcal{N}$ nodos del sistema, *restringida* para sesiones donde $t_{i,2}+\tau \geq t_{i+1,1}$, es decir, cuando un nodo llega antes de la expiración de su membresía; si ésta termina, se considera que la siguiente llegada del nodo, si la hay, es la de uno nuevo.

La Tabla 8.1 muestra los costos particulares de ancho de banda por mantenimiento de cada una de las estrategias de redundancia revisadas hasta ahora. Cada una de esas fórmulas está basada en la Ec. 8.3, con $\mathcal{D} = \mathcal{F}M$, y un factor de redundancia o estiramiento k propio de cada tipo de redundancia. En secciones posteriores haremos referencia a ella para hablar de los resultados obtenidos en la experimentación. En este sentido, las variables M, $\mathcal{F}$ y $\mathcal{N}$ tienen el mismo significado presentado en la Tabla 7.2.

Tabla 8.1: *Costos en ancho de banda debido a mantenimiento.*

Estrategia	**Almacenamiento promedio por nodo,** $\bar{d}$ [bytes]	**Ancho de banda por mantenimiento promedio por nodo,** $\bar{b}(\tau)$ [bytes/s]
RB	$k_B \frac{\mathcal{F}M}{\mathcal{N}}$	$2\,\frac{k_B(A,a(\tau),m)\mathcal{F}M}{T(\tau)}$
RS	$k_S \frac{\mathcal{F}M}{\mathcal{N}}$	$2\,\frac{k_S(A,a(\tau))\mathcal{F}M}{T(\tau)}$
RIDA	$k_I \frac{\mathcal{F}M}{\mathcal{N}}$	$2\,\frac{k_I(A,a(\tau),m)\mathcal{F}Mm}{T(\tau)}$
RH	$(1+k_I)\frac{\mathcal{F}M}{\mathcal{N}}$	$2\,\frac{(1+k_I(A,a(\tau),m))\mathcal{F}M}{T(\tau)}$
RCR	$k_R \frac{\mathcal{F}M}{\mathcal{N}}$	$2\,\frac{k_R(A,a(\tau),m)\mathcal{F}M}{T(\tau)}\,\frac{m^2}{m^2-m+1}$

En § 2.3.2 presentamos un versión de fuente digital que usa códigos lineales aleatorios, de igual manera que los códigos de red en § 2.3.2. Es por eso que consideramos que los resultados de [23] (§ 2.3.3) para códigos de red pueden extenderse —al menos bajo un escenario de simulación como el nuestro— para una fuente digital. En adelante, sólo haremos alusión a los resultados que tenemos para la estrategia RCR.

8.2. Trazas

Para nuestros experimentos utilizamos, al igual que [75] y [23] una serie de archivos que contienen información que rastrea la entrada y salida de nodos en diferentes sistemas P2P. A ese tipo de archivos les llamamos *trazas*; en la Tabla 8.2 se describen cada una de las ellas.

Todas las trazas de la Tabla 8.2 están compuestas de líneas, cada una de las cuales contiene información sobre un solo nodo: un identificador, el número de entradas y salidas del sistema, y pares de estampillas de tiempo $(t_{i,1}, t_{i,2})$ que indican los tiempos de inicio y fin de cada sesión. Cada valor está separado por espacios en blanco. Esto es

[Identificador del nodo] [Número de sesiones] $t_{1,1}$ $t_{1,2}$ $t_{2,1}$ $t_{2,2}$ [$\cdots$] $t_{n,1}$ $t_{n,2}$

Figura 8.2: *Disponibilidad promedio de los nodos $a(\tau)$.*

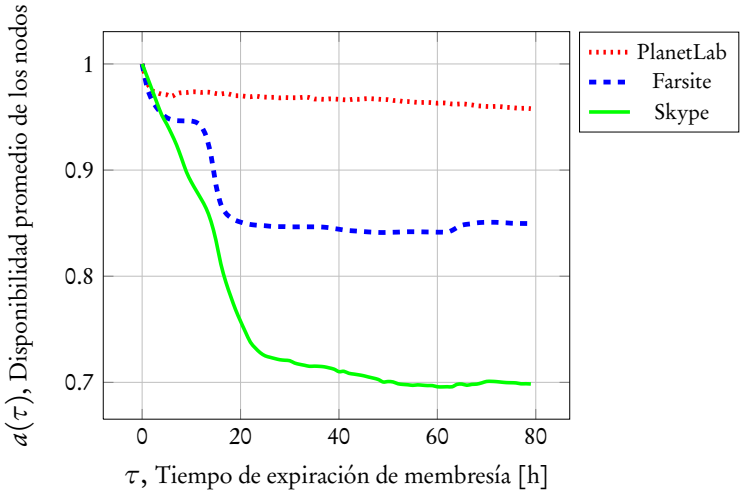


Figura 8.3: *Miembros del sistema.*

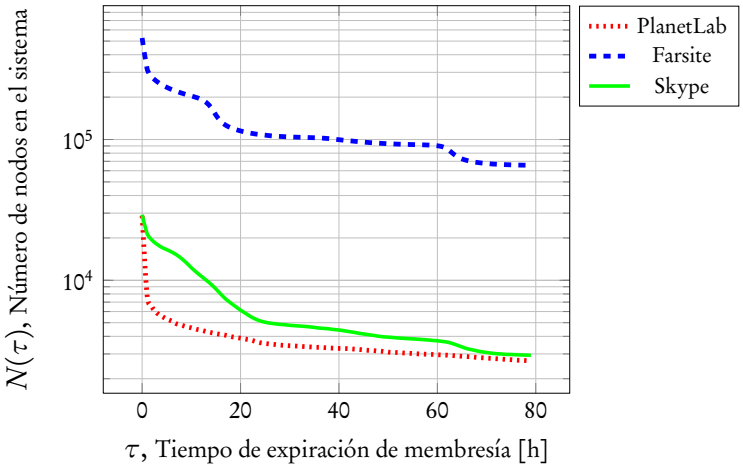


Figura 8.4: *Tasa de fallas por hora.*

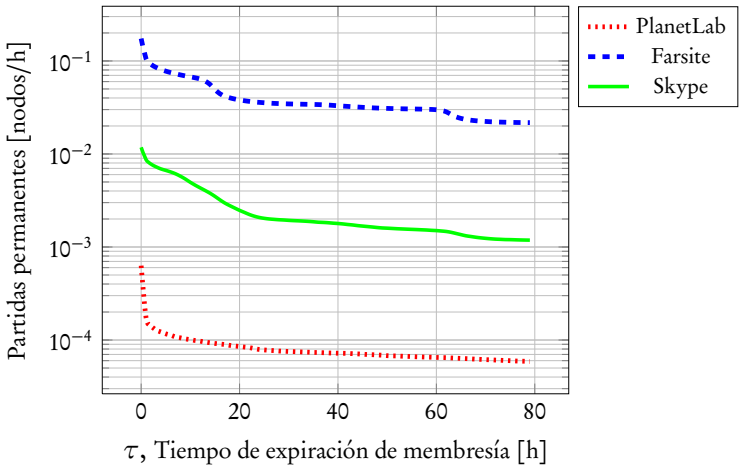


Tabla 8.2: *Trazas*.

Traza	Descripción	Número de nodos	Duración [días]	Promedio de nodos activos
PlanetLab [87]	Es el resultado de monitorear cada 15 minutos el estado de un grupo de nodos del sistema PlanetLab desde enero de 2004 hasta junio de 2005. Las pruebas de monitoreo se hicieron entre todos los pares de nodos posibles.	669	527.46	299
Farsite [14]	Más de 50,000 computadoras de escritorio dentro de las oficinas de Microsoft fueron monitoreadas cada hora durante 35 días a partir del 6 de julio de 1999.	51,547	35.00	41,969
Skype [36]	Un conjunto de más de 2,000 participantes en la red Skype fueron monitoreados cada 30 minutos durante un mes a partir del 12 de septiembre de 2005.	2,081	28.68	690

Procesamos cada una de las trazas para obtener la disponibilidad promedio de los nodos $a(\tau)$ variando el tiempo de expiración de membresía τ. En la Fig. 8.2 se muestran los resultados. Allí puede verse que los nodos de PlanetLab son más estables que los de Farsite y, a su vez, estos lo son más que los de Skype. En todas las curvas se ve una caída de $a(\tau)$ en la medida que aumenta τ. Hablamos de estabilidad cuando el sistema presenta disponibilidades promedio por nodo cercanas a 1 y éstas muestran ligeras variaciones en función de τ.

Cuando τ crece, el sistema tarda más en reconocer que un nodo partió permanentemente, y la disponibilidad promedio de los nodos baja porque hay más sesiones temporales con espacios de tiempo más largos entre ellas, a la vez que el tiempo de membresía t_m también crece. Esto también puede inferirse a partir de la Ec. 8.2 cuando $t_m \to \infty$.

Los resultados de la Fig. 8.2 servirán para calcular, con base en la Ec. 8.3, el ancho de banda promedio por nodo $\overline{b}$, de cada una de las estrategias de redundancia de nuestro interés, y que es necesario para mantener la redundancia existente en el sistema.

La Fig. 8.3 muestra cómo cambia el número de miembros del sistema en función del tiempo de expiración de membresía. En el caso extremo $\tau = 0$ hay mayor número de sesiones porque todas las partidas de los nodos se consideran como permanentes de acuerdo con la heurística. El caso inverso sucede cuando $\tau \to \infty$.

Observamos que aunque los nodos que pertenecen a Skype tienen menor disponibilidad en promedio que los de Farsite (Fig. 8.2), esto no implica necesariamente que la tasa de fallas de Skype sea mayor (Fig. 8.4). Aunque la intuición dicta que mayor disponibilidad promedio de los nodos implica menos partidas permanentes, esa discrepancia indica que hay una mayor cantidad de tiempo usado entre partidas temporales en los nodos de Skype que en los de Farsite. No obstante, PlanetLab sigue siendo el sistema con los nodos más estables, también al tener la menor tasa de fallas en comparación con los demás.

8.3. Experimentación

En esta sección utilizamos las fórmulas de ancho de banda promedio por nodo $\overline{b}$ para mantener la redundancia, mostradas en la Tabla 8.1. Graficamos $\overline{b}(\tau)$ en función del tiempo de expiración de membresía τ para cada una de las estrategias de redundancia de nuestro interés. Para cada gráfica fijamos los valores presentados en la Tabla 8.3.

Tabla 8.3: *Variables para nuestros escenarios de evaluación.*

Variable	Valor
A	0.9999
M	700 MB por objeto
m	100 unidades
$\mathcal{F}$	1000 objetos

En las Fig. 8.5, 8.6, 8.7, 8.8 y 8.9 mostramos el impacto en ancho de banda para mantener la redundancia perdida debido a fallas —i.e., partidas permanentes— en los nodos del sistema. En todas ellas observamos el mismo patrón de comportamiento:

- Como resulta lógico, conforme aumenta τ, disminuye $\overline{b}$ en todos los casos, porque también disminuye la disponibilidad promedio por nodo.
- El orden en el que se presentan las curvas. De arriba a abajo, Skype, Farsite y PlanetLab. Donde, mientras más abajo, menor $\overline{b}$ es necesario. Esta disposición de las curvas nos habla de que la disponibilidad promedio de los nodos, tomando en cuenta la Fig. 8.2, es el parámetro que más afecta el uso de ancho de banda.

En las Figs. 8.10, 8.11 y 8.12 se muestran comparaciones entre todas las estrategias revisadas para cada traza. También entre ellas se observa la misma disposición de las curvas. Claramente, RIDA es la que peor desempeño muestra cuando existen fallas en el sistema. En cambio, RCR ofrece el mejor desempeño de todas las estrategias. RIDA tiene una desventaja de hasta tres órdenes de magnitud con respecto a RCR.

Con RIDA es muy costoso restaurar un objeto de información porque por cada disperso perdido se requiere la transferencia de m de ellos a un nodo recién llegado (§ 3.4.2 y Fig. 3.14), para luego construir el objeto original, y de éste el disperso faltante. RCR ofrece las mejores ventajas en ancho de banda porque actúan con los límites teóricos relacionados con el flujo de la información descritos en § 2.3.3 y la Fig. 2.12. Lo mismo podemos concluir para RFD por las razones antes expuestas.

La estrategia híbrida presentó el segundo mejor desempeño. Esto se debe a que no es necesario enviar m unidades para recuperar un disperso, sino que gracias al objeto de información extra en el sistema, a partir de éste es posible crear el disperso perdido (§ 3.4.3 y Fig. 3.17). De este modo, son sólo una fracción de M/m bytes del objeto original los que se necesita transmitir para recuperar un disperso.

Figura 8.5: $\overline{b}(\tau)$ *para RB.*

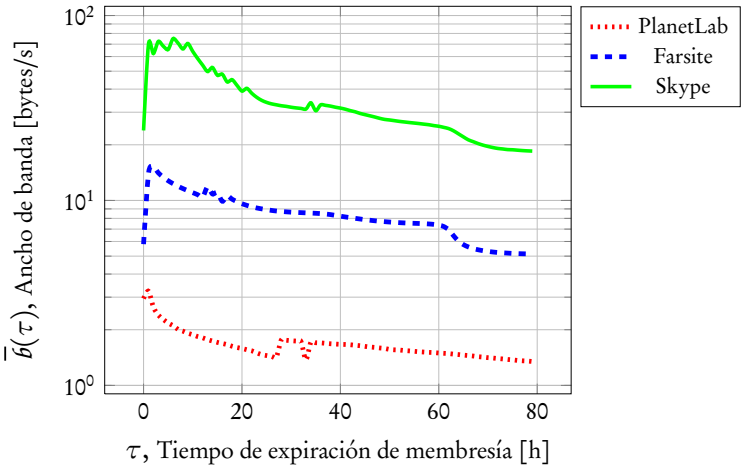


Figura 8.6: $\overline{b}(\tau)$ *para RS.*

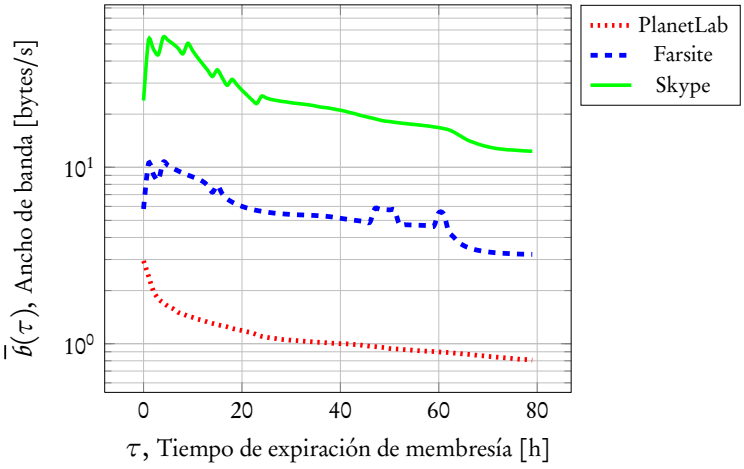


Figura 8.7: $\overline{b}(\tau)$ *para RIDA.*

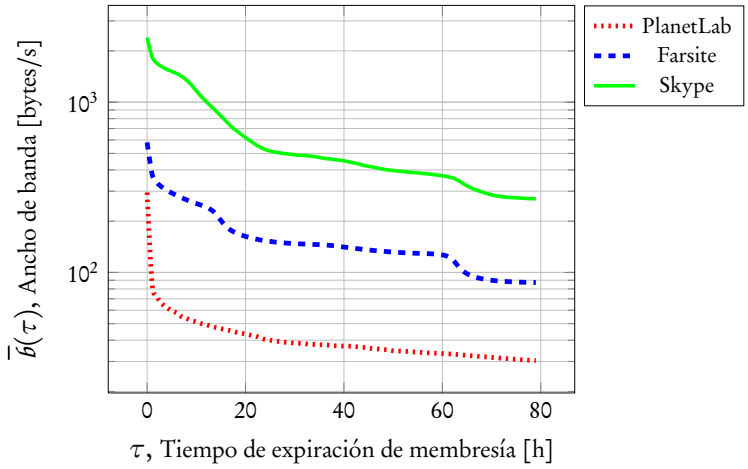


Figura 8.8: $\overline{b}(\tau)$ *para RCR.*

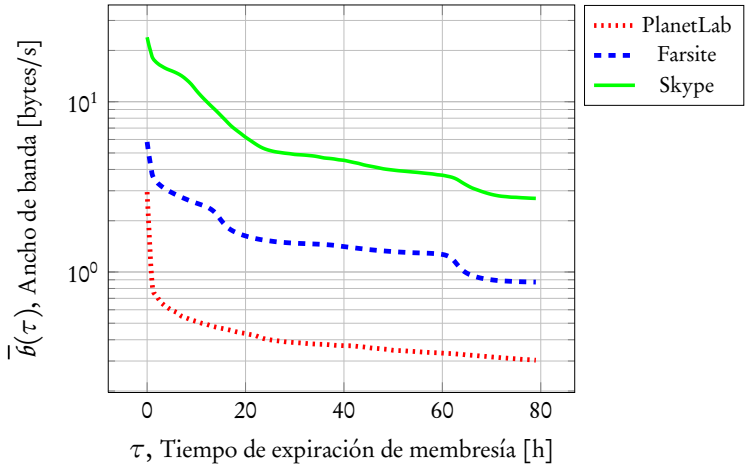


Figura 8.9: $\overline{b}(\tau)$ *para RH.*

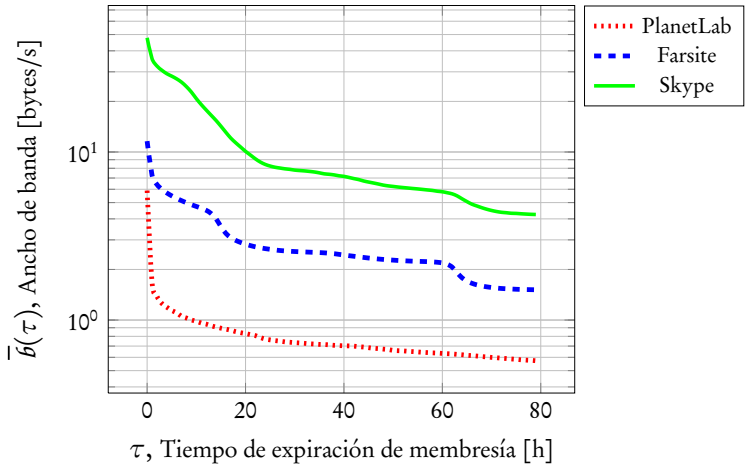


Figura 8.10: *PlanetLab.*

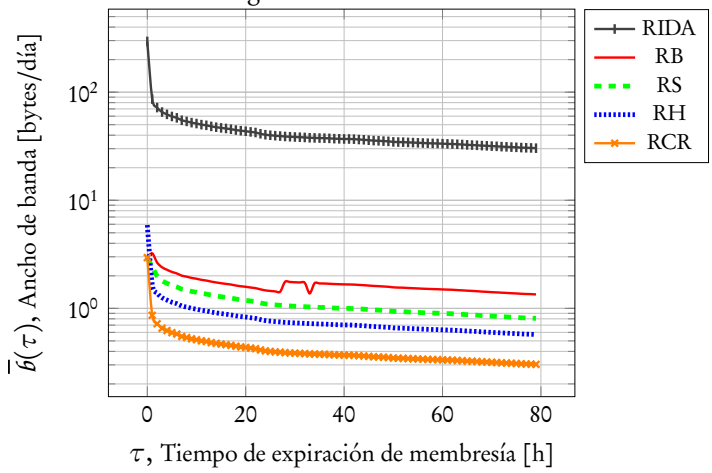


Figura 8.11: *Skype.*

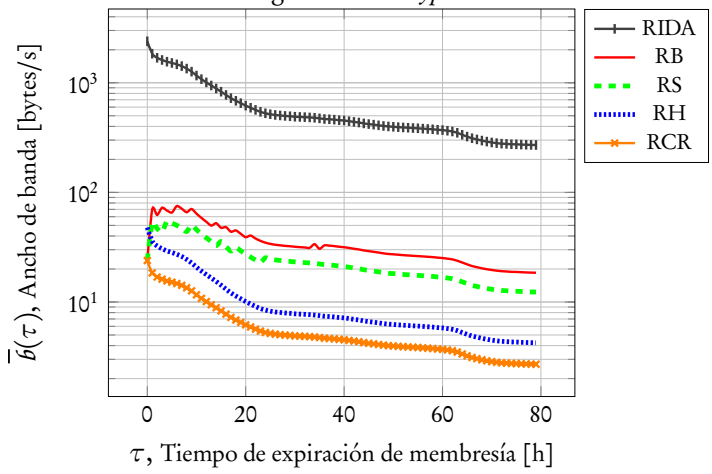
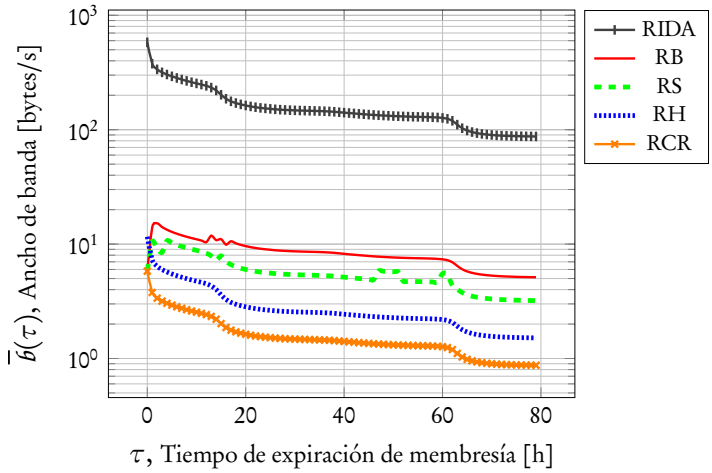


Figura 8.12: *Microsoft.*



Parte III

Discusión final

Capítulo 9

Conclusiones y Trabajo Futuro

El objetivo general de este trabajo de investigación fue evaluar un conjunto representativo de estrategias para redundar información, con el fin de ofrecer al lector elementos suficientes para ayudarle a decidir cuál estrategia es más apropiada para la implementación de un sistema de almacenamiento P2P. En este sentido, y luego de hacer una revisión de la literatura relevante en el tema, consideramos apropiado establecer los siguientes objetivos particulares:

- Realizar un compendio ordenado y detallado de los resultados más importantes encontrados en relación con la evaluación de estrategias de redundancia en sistemas de almacenamiento P2P.
- Agregar resultados experimentales que complementen y faciliten la comprensión del comportamiento de cada estrategia. Por ejemplo, al proponer nuevas gráficas a las existentes en la literatura.
- Proponer una nomenclatura adecuada para la temática.
- Concluir con la realización de una guía que intersecte todas las estrategias evaluadas.

Con base en esos objetivos, a continuación listamos las que juzgamos como principales aportaciones de este trabajo de investigación:

- Realizamos un examen más extenso y detallado (§ 6, § 7 y § 8) de los resultados obtenidos por [9] y [23]. Donde incluimos más estrategias de redundancia de información en la evaluación de la redundancia, almacenamiento y ancho de banda. Así también, cálculamos intervalos de confianza para los valores de almacenamiento promedio por nodo $\overline{d}$, y creamos una nomenclatura consistente a lo largo de este documento.
- Ofrecimos una representación genérica (§ 3.3), por etapas, del proceso de almacenar información de manera distribuida. Donde es de particular importancia la que llamamos Etapa M-3, porque en ella se realiza el mantenimiento de la redundancia de información luego de la detección de pérdida de información en el sistema.
- Nuestro trabajo, en su conjunto, puede considerarse como un *survey* sobre los sistemas de almacenamiento P2P, en general, y de las estrategias de redundancia de información, en particular. Cabe destacar que a diferencia de otros trabajos de investigación (§ 4), en este incluimos resultados para una estrategia de redundancia basada en fuentes digitales, y una serie de comparaciones más completa para todas las estrategias (§ 6.5, § 7.5 y § 8.3). A lo anterior se añade una extensa bibliografía especializada en el tema.

- Presentamos una *guía* (Tabla 9.1), fundamentada en nuestro resultados, de las circunstancias bajo las cuales recomendamos, o no, el uso de cada estrategia de redundancia revisada aquí. En seguida, presentamos esa guía, cuyos puntos pueden considerarse como las conclusiones más importantes de este trabajo de investigación.

Guía para elegir una estrategia de redundancia de información

Luego de analizar los resultados de nuestra investigación, llegamos a las siguientes conclusiones en lo que respecta al desempeño de diferentes estrategias de redundancia de información en sistemas de almacenamiento P2P.

- Las estrategias que usan códigos —RIDA, RCR, RFD— tienen mejor desempeño en redundancia (Etapa P-2, § 3.3.1, y Fig. 6.9) y almacenamiento (Etapa D-2, § 3.3.2, y Fig. 7.17) de información que las estrategias que no los usan —RS, RB—. Lo anterior se manifiesta en mayor medida cada vez que la disponibilidad de los nodos del sistema tiende a disminuir.

- La estrategia híbrida RH —RIDA más RS con $k_S = 1$— tiene costos similares en almacenamiento que RIDA (Fig. 7.17), pero tiene un desempeño considerablemente mejor en ancho de banda por mantenimiento, hasta por dos órdenes de magnitud (Figs. 8.10, 8.11 y 8.12).

- Mientras más alta sea la tasa de fallas (Fig. 8.4, § 3.3.3), más ancho de banda se requiere para mantener los niveles de redundancia del sistema. Esta tendencia prevalece en todas la estrategias de redundancia estudiadas. Esto también implica que las condiciones de *churn* afectan negativamente el desempeño del sistema en todas las etapas definidas en § 3.3. En este sentido, confirmamos la principal conclusión de [12]: un *sistema de almacenamiento P2P que ofrezca alta disponibilidad de la información, en un ambiente con una alta dinámica de nodos, donde el almacenamiento escale con la cantidad de nodos*, es todavía problema abierto.

- Si bien un esquema híbrido de redundancia ofrece ventajas en ancho de banda, también agrega complejidad en el diseño del sistema porque deben administrarse varios tipos de redundancia. Por ejemplo, antes de recuperar la redundancia perdida, el sistema debe realizar diversas verificaciones que conllevan uso de recursos de cómputo y comunicaciones que finalmente pueden afectar la calidad del servicio, sobre todo bajo condiciones de red con baja disponibilidad en los nodos. Sin embargo, la copia extra con la estrategia RH afecta más el almacenamiento con respecto a RIDA en la medida que los nodos son más estables (Fig 7.16).

- Las fuentes digitales y los códigos de red pueden utilizarse como cualquier otra estrategia de códigos de borrado tradicional —e.g., Reed-Solomon, IDA—, con las ventajas agregadas en complejidad de códificación y decodificación, y uso de ancho de banda para mantenimiento.

- Los mecanismos para mantener los niveles de redundancia de información (Etapa M-3, § 3.3.3) ante fallas en el sistema son muy particulares de cada estrategia de redundancia.

La Tabla 9.1 resume nuestros resultados y su propósito es servir como guía para el diseñador de un sistema de almacenamiento P2P, de manera tal que pueda tomar una mejor decisión sobre cuál estrategia de redundancia de información elegir en función de ciertos parámetros clave.

Tabla 9.1: *Guía para elegir una estrategia de redundancia de información.*

Estrategia	Cuándo usarla
RB	Su uso no es recomendable en general.
RS	Cuando el almacenamiento no sea una restricción crítica para el diseño del sistema. Cuando los nodos tengan altos niveles de disponibilidad en promedio.
RIDA	Cuando el uso de almacenamiento sea crítico, y no así el ancho de banda de mantenimiento. En este sentido, el sistema donde se quiera implantar debe ofrecer altos niveles de disponibilidad promedio por nodo ($a > 0.9$).
RFD	Cuando el ancho de banda de mantenimiento sea de importancia crítica, así como el uso de CPU. El sistema donde se quiera implantar puede ser *muy* inestable ($a < 0.5$).
RCR	Cuando el ancho de banda de mantenimiento sea de importancia crítica. El sistema donde se quiera implantar puede ser inestable ($a < 0.8$).
RH	Su uso es recomendable para un amplio rango de restricciones en ancho de banda y almacenamiento. No así cuando la disponibilidad de los nodos sea muy alta ($a > 0.95$), debido al costo de almacenar una copia extra de información.

Trabajo futuro

En lo que sigue presentamos una serie de recomendaciones de posible trabajo futuro que puede realizarse a partir del nuestro. También hacemos hincapié en las que consideramos aún preguntas abiertas.

- *Reconciliar contenido* en un sistema de almacenamiento que utiliza códigos tipo IDA, significa verificar que para reconstruir información no se reciben dispersos repetidos; de ser así, entonces el sistema inicia un proceso de reexpedición de información que logre reunir un conjunto adecuado de ellos. Esto se traduce en retrasos y mayor complejidad en el diseño del sistema. En la práctica, las fuentes digitales crean una muy alta cantidad de dispersos distintos entre sí, de modo que tienen la capacidad de evitar, con alta probabilidad, el problema de reconciliación de contenido que existe con otras técnicas de codificación. Consideramos que aún hay trabajo por hacer para explorar los efectos positivos en ancho de banda y latencia del sistema cuando se evita la reconciliación de contenido. Para esto es posible que se necesite hacer simulaciones más precisas, *paquete por paquete* en un entorno de almacenamiento P2P.

- Los resultados teóricos de [23] en relación con los códigos de red nos permitieron estimar el ancho de banda que pueden utilizar en función de la disponibilidad de los nodos. Sin embargo, no encontramos evidencia experimental que sustente dicho resultado, como sí ocurre con los modelos probabilisticos obtenidos en [9] acerca de la disponibilidad de la información (Ecs. (6.6) y (7.1)). La obtención de esa evidencia con códigos de red es un trabajo que queda por hacer.

- En nuestro trabajo utilizamos archivos —trazas— que rastrean las entradas y salidas de nodos participantes en sistemas reales: Skype, PlanetLab y Farsite. También existen otras trazas de acceso público que pueden presentar comportamientos más complejos o diversos. No obstante, consideramos que

es posible crear trazas artificiales, las cuales pueden *categorizarse* por su impacto en el uso de recursos para cada estrategia de redundancia de información.

- A lo largo de este trabajo de investigación asumimos que las fallas en los nodos se daban de manera independiente e idénticamente distribuidas. Aunque esta es una buena asunción en general, en la realidad las fallas pueden correlacionarse. Por ejemplo, el comportamiento de un subconjunto de nodos del sistema puede depender del huso horario de su ubicación geográfica —e.g., cuando lo nodos parten a la media noche— y de la topología de la red a la que pertenecen —e.g., así la partida permanente de un nodo puede afectar a sus vecinos.

- La evaluación de otra estrategia de redundancia híbrida. Por ejemplo, una que incluya redundancia simple —con una copia o más— de la mano con una fuente digital, o bien, con códigos de red. Es probable que se obtengan mejores resultados en el uso de ancho de banda para mantenimiento.

- Otro aspecto a considerar, es la evaluación de la *latencia* para cada una de las estrategias de redundancia. En nuestro contexto, la latencia puede considerarse como la suma de los tiempos de retardo en la entrega de la información. Los retardos estarían causados, entre otros factores, por la condiciones de uso de la red, la ubicación física y lógica de los nodos, los protocolos de transporte, el control de congestión de la red, el tamaño de los datos —e.g., número de símbolos por archivo—, las aplicaciones P2P encargadas de mantener la procesar, distribuir y mantener la información.

- Al tiempo de realización de este trabajo de investigación, tal como mencionamos en § 3.1.4, los simuladores P2P aún ofrecen condiciones inadecuadas para el desarrollo de experimentos. Sin embargo, en el futuro esas condiciones pueden mejorar para realizar experimentos más detallados sobre estrategias de redundancia y su efecto sobre diferentes parámetros de desempeño en el contexto de los sistemas de almacenamiento distribuido. Es así que proponemos tomar en cuenta una nueva revisión del estado del arte de los simuladores de sistemas P2P.

Bibliografía

[1] A Conversation with Jim Gray. *Queue 1*, 4 (2003), 8–17.

[2] ACEDAŃSKI, S., DEB, S., MÉDARD, M., AND KOETTER, R. How good is random linear coding based distributed networked storage. In *NetCod* (2005).

[3] AHLSWEDE, R., CAI, N., LI, S., AND YEUNG, R. Network information flow. *Information Theory, IEEE Transactions on 46*, 4 (2000), 1204–1216.

[4] AMAR, J. G. The Monte Carlo Method in Science and Engineering. *Computing in Science and Engg. 8*, 2 (2006).

[5] ANDERSEN, D. G. Overlay Networks: Networking on Top of the Network. ACM Computing Reviews: Hot Topic Essay, January 2005. http://www.reviews.com/hottopic/hottopic_essay_01.cfm.

[6] ANDERSON, D., COBB, J., KORPELA, E., LEBOFSKY, M., AND WERTHIMER, D. SETI@ home: an experiment in public-resource computing. *Communications of the ACM 45*, 11 (2002), 56–61.

[7] ANDROUTSELLIS-THEOTOKIS, S., AND SPINELLIS, D. A Survey of Peer-to-Peer Content Distribution Technologies. *ACM Computing Surveys (CSUR) 36*, 4 (2004), 335–371.

[8] BAUMGART, I., HEEP, B., AND KRAUSE, S. OverSim: A Flexible Overlay Network Simulation Framework. In *Proceedings of 10th IEEE Global Internet Symposium (GI '07) in conjunction with IEEE INFOCOM 2007, Anchorage, AK, USA* (May 2007).

[9] BHAGWAN, R., MOORE, D., SAVAGE, S., AND VOELKER, G. M. Replication Strategies for Highly Available Peer-to-Peer Storage Systems. In *Future Directions in Distributed Computing* (2003), pp. 153–158.

[10] BHAGWAN, R., SAVAGE, S., AND VOELKER, G. M. Understanding availability. In *IPTPS* (2003), pp. 256–267.

[11] BIRMAN, K. P. Technology Challenges for Virtual Overlay Networks. *IEEE Transactions on Systems, Man, and Cybernetics, Part A 31*, 4 (2001), 319–327.

[12] BLAKE, C., AND RODRIGUES, R. High availability, scalable storage, dynamic peer networks: pick two. In *HOTOS'03: Proceedings of the 9th conference on Hot Topics in Operating Systems* (Berkeley, CA, USA, 2003), USENIX Association, pp. 1–1.

[13] BLANCO, R., AHMED, N., HADALLER, D., SUNG, L. G. A., LI, H., AND SOLIMAN, M. A. A survey of data management in peer-to-peer systems. Tech. rep., David R. Cheriton School of Computer Science, University of Waterloo, Canada, June 2006.

[14] BOLOSKY, W., DOUCEUR, J., ELY, D., AND THEIMER, M. Feasibility of a serverless distributed file system deployed on an existing set of desktop PCs. *Proceedings of the 2000 ACM SIGMETRICS international conference on Measurement and modeling of computer systems* (2000), 34–43.

[15] BRAY, G. GnuMap Project, June 2007. http://home.comcast.net/ gregory.bray.

[16] BYERS, J. W., LUBY, M., MITZENMACHER, M., AND REGE, A. A digital fountain approach to reliable distribution of bulk data. *SIGCOMM Comput. Commun. Rev. 28*, 4 (1998), 56–67.

[17] CHAWATHE, Y., RATNASAMY, S., BRESLAU, L., LANHAM, N., AND SHENKER, S. Making gnutella-like P2P systems scalable. *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications* (2003), 407–418.

[18] CHEVANCE, R. J. *Server Architectures: Multiprocessors, Clusters, Parallel Systems, Web Servers, and Storage Solutions*. Digital Press, 2005.

[19] COHEN, B. Incentives Build Robustness in BitTorrent. *Workshop on Economics of Peer-to-Peer Systems 6* (2003).

[20] CORMEN, T. H., LEISERSON, C. E., RIVEST, R. L., AND STEIN, C. *Introduction to Algorithms*, 2nd ed. The MIT Press, 2001.

[21] DABEK, F. *A Distributed Hash Table*. PhD thesis, Cambridge, MA, USA, 2006. Adviser-Kaashoek„ M. Frans and Adviser-Morris„ Robert T.

[22] DABEK, F., KAASHOEK, M. F., KARGER, D., MORRIS, R., AND STOICA, I. Wide-area cooperative storage with CFS. *SIGOPS Oper. Syst. Rev. 35*, 5 (2001), 202–215.

[23] DIMAKIS, A., GODFREY, P., WAINWRIGHT, M., AND RAMCHANDRAN, K. Network coding for distributed storage systems. *INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE* (6-12 May 2007), 2000–2008.

[24] DIMAKIS, A. G., AND RAMCHANDRAN, K. *Network Coding for Distributed Storage in Wireless Networks*. Springer Verlag, 2007, ch.Ñetworked Sensing Information and Control.

[25] DOVAL, D., AND O'MAHONY, D. Overlay Networks: A Scalable Alternative for P2P. *IEEE Internet Computing 7*, 4 (2003), 79–82.

[26] FOSTER, I., AND IAMNITCHI, A. On Death, Taxes, and the Convergence of Peer-to-Peer and Grid Computing. In *IPTPS* (2003), pp. 118–128.

[27] FOSTER, I., AND KESSELMAN, C. *The Grid 2: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003.

[28] FOSTER, I., KISHIMOTO, H., SAVVA, A., BERRY, D., DJAOUI, A., GRIMSHAW, A., HORN, B., MACIEL, F., SIEBENLIST, F., SUBRAMANIAM, R., TREADWEL, J., AND REICH, J. V. The Open Grid Services Architecture, Version 1.0. Tech. rep., Global Grid Forum, January 29 2005.

[29] FRAGOULI, C., AND WIDMER, J. Network coding: an instant primer. *ACM SIGCOMM Computer Communication Review 36*, 1 (2006), 63–68.

[30] GHODSI, A. *Distributed k-ary System: Algorithms for Distributed Hash Tables*. PhD dissertation, KTH—Royal Institute of Technology, Stockholm, Sweden, Dec. 2006.

[31] GIBSON, G. A., AND METER, R. V. Network Attached Storage Architecture. *Commun. ACM 43*, 11 (2000), 37–45.

[32] GKANTSIDIS, C., AND RODRIGUEZ, P. Network coding for large scale content distribution. *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE 4* (2005).

[33] GLOBUS. The Globus Alliance Home Page, June 2007. http://www.globus.org.

[34] GNUTELLA. The Gnutella Home Page, June 2007. http://www.gnutella.com.

[35] GRAY, J., AND SIEWIOREK, D. P. High-availability computer systems. *Computer 24*, 9 (1991), 39–48.

[36] GUHA, S., DASWANI, N., AND JAIN, R. An Experimental Study of the Skype Peer-to-Peer VoIP System. In *Proceedings of The 5th International Workshop on Peer-to-Peer Systems (IPTPS '06)* (Santa Barbara, CA, February 2006), pp. 1 – 6.

[37] GUMMADI, P. K., SAROIU, S., AND GRIBBLE, S. D. A measurement study of napster and gnutella as examples of peer-to-peer file sharing systems. *Computer Communication Review 32*, 1 (2002), 82.

[38] GUPTA, A., LISKOV, B., AND RODRIGUES, R. One hop lookups for peer-to-peer overlays. In *HOTOS'03: Proceedings of the 9th conference on Hot Topics in Operating Systems* (Berkeley, CA, USA, 2003), USENIX Association, pp. 2–2.

[39] HARRISON, A., AND TAYLOR, I. Service-oriented middleware for hybrid environments. In *ADPUC '06: Proceedings of the 1st international workshop on Advanced data processing in ubiquitous computing (ADPUC 2006)* (New York, NY, USA, 2006), ACM Press, p. 2.

[40] HASAN, R., ANWAR, Z., YURCIK, W., BRUMBAUGH, L., AND CAMPBELL, R. A Survey of Peer-to-Peer Storage Techniques for Distributed File Systems. In *ITCC '05: Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'05) - Volume II* (Washington, DC, USA, 2005), IEEE Computer Society, pp. 205–213.

[41] HUFFMAN, W. C., AND PLESS, V. *Fundamentals of error-correcting codes*. Cambridge Univ. Press, 2003.

[42] KARGER, D., LEHMAN, E., LEIGHTON, T., PANIGRAHY, R., LEVINE, M., AND LEWIN, D. Consistent hashing and random trees: distributed caching protocols for relieving hot spots on the World Wide Web. *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing* (1997), 654–663.

[43] KAZAA. The Kazaa Home Page, June 2007. http://www.kazaa.com.

[44] KLEINROCK, L. *Theory, Volume 1, Queueing Systems*. Wiley-Interscience, 1975.

[45] KRAKOWIAK, S. ObjectWeb Consortium: Open Source Middleware, June 2007. http://middleware.objectweb.org/.

[46] LIN, W. K., CHIU, D. M., AND LEE, Y. B. Erasure code replication revisited. In *P2P '04: Proceedings of the Fourth International Conference on Peer-to-Peer Computing* (Washington, DC, USA, 2004), IEEE Computer Society, pp. 90–97.

[47] LINGA, P., GUPTA, I., AND BIRMAN, K. A churn-resistant peer-to-peer web caching system. In *SSRS '03: Proceedings of the 2003 ACM workshop on Survivable and self-regenerative systems* (New York, NY, USA, 2003), ACM, pp. 1–10.

[48] LUBY, M. LT codes. *Foundations of Computer Science, 2002. Proceedings. The 43rd Annual IEEE Symposium on* (2002), 271–280.

[49] MACKAY, D. J. C. *Information Theory, Inference & Learning Algorithms*. Cambridge University Press, June 2002.

[50] MALKHI, D., NAOR, M., AND RATAJCZAK, D. Viceroy: a scalable and dynamic emulation of the butterfly. In *PODC '02: Proceedings of the twenty-first annual symposium on Principles of distributed computing* (New York, NY, USA, 2002), ACM Press, pp. 183–192.

[51] MAYMOUNKOV, P. Online codes. *Research Report TR2002-833, New York University, Nov* (2002).

[52] MAYMOUNKOV, P., HARVEY, N., AND LUN, D. Methods for efficient network coding. In *44th Annual Allerton Conference on Communication, Control, and Computing* (2006).

[53] MAYMOUNKOV, P., AND MAZIÈRES, D. Kademlia: A peer-to-peer information system based on the xor metric. In *IPTPS '01: Revised Papers from the First International Workshop on Peer-to-Peer Systems* (London, UK, 2002), Springer-Verlag, pp. 53–65.

[54] MENN, J. *All the Rave: The Rise and Fall of Shawn Fanning's Napster*. Crown Business, Random House Inc., New York, USA, 2003.

[55] MILOJICIC, D. S., KALOGERAKI, V., LUKOSE, R., K., N., PRUYNE, J., RICHARD, B., ROLLINS, S., AND XU, Z. Peer-to-Peer Computing. Tech. rep., Hewlett-Packard Company, USA, March 2006.

[56] MONNERAT, L. R., AND AMORIM, C. L. D1HT: A Distributed One Hop Hash Table. Tech. rep., In Proc of the 20th IEEE Intl Parallel and Distributed Processing Symposium (IPDPS), 2005.

[57] MOON, T. K. *Error Correction Coding: Mathematical Methods and Algorithms*. Wiley-Interscience, 2005.

[58] MOORE, F. G. Storage Facts, Figures, Estimates and Rules of Thumb. Horison Information Strategies, June 2006. http://www.horison.com/StorageFactsandFigures.doc.

[59] MOREIRA, J. C., AND FARREL, P. G. *Essentials of Error-Correcting Coding*. John-Wiley and Sons, 2006.

[60] NAICKEN, S., BASU, A., LIVINGSTON, B., AND RODHETBHAI, S. A Survey of Peer-to-Peer Network Simulators. In *Proceedings of The Seventh Annual Postgraduate Symposium, Liverpool, UK* (2006).

[61] NAICKEN, S., LIVINGSTON, B., BASU, A., RODHETBHAI, S., WAKEMAN, I., AND CHALMERS, D. The state of peer-to-peer simulators and simulations. *SIGCOMM Comput. Commun. Rev. 37*, 2 (2007), 95–98.

[62] NAPSTER. The Napster Home Page, June 2007. http://www.napster.com.

[63] ORAM, A., Ed. *Peer-to-Peer: Harnessing the Power of Disruptive Technologies*. O'Reilly & Associates, Inc. Sebastopol, CA, USA, February 2001.

[64] P2PRG. Peer-to-Peer Research Group: Charter, July 2007. http://www.irtf.org.

[65] PATTERSON, D. A. Latency lags bandwith. *Commun. ACM 47*, 10 (2004), 71–75.

[66] PATTERSON, D. A., AND HENNESSY, J. L. *Computer Organization and Design (4th ed.): The Hardware/Software Interface*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2006.

[67] PLACEK, M., AND BUYYA, R. A Taxonomy of Distributed Storage Systems. Tech. rep., University of Melbourne, Grid Computing and Distributed Systems Laboratory, July 2006.

[68] PLANETSIM. Object Oriented Simulation Framework for Overlay Networks, September 2007. http://astdeim.urv.cat/trac/planetsim/wiki/PlanetSim.

[69] PLAXTON, C., RAJARAMAN, R., AND RICHA, A. Accessing Nearby Copies of Replicated Objects in a Distributed System. *Proceedings of the Symposium of Parallel Algorithms and Architectures (SPAA'97)* (1997), 311–320.

[70] RABIN, M. Efficient dispersal of information for security, load balancing, and fault tolerance. *Journal of the ACM (JACM) 36*, 2 (1989), 335–348.

[71] RATNASAMY, S., FRANCIS, P., HANDLEY, M., KARP, R., AND SCHENKER, S. A scalable content-addressable network. In *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications* (New York, NY, USA, 2001), ACM Press, pp. 161–172.

[72] RHEA, S., GODFREY, B., KARP, B., KUBIATOWICZ, J., RATNASAMY, S., SHENKER, S., STOICA, I., AND YU, H. OpenDHT: a public DHT service and its uses. *Proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for computer communications* (2005), 73–84.

[73] RIPEANU, M. Peer-to-peer architecture case study: Gnutella network. *Proceedings of International Conference on Peer-to-peer Computing 101* (2001).

[74] RIPEANU, M., AND FOSTER, I. Mapping the Gnutella Network: Macroscopic Properties of Large-Scale Peer-to-Peer Systems. *First International Workshop on Peer-to-Peer Systems (IPTPS) 68* (2002).

[75] RODRIGUES, R., AND LISKOV, B. High Availability in DHTs: Erasure Coding vs. Replication. In *IPTPS* (2005), pp. 226–239.

[76] ROMAN, S. *Coding and information theory*. Springer-Verlag New York, Inc., New York, NY, USA, 1992.

[77] ROSS, S. M. *A Course in Simulation*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1990.

[78] ROUSSOPOULOS, M., BAKER, M., ROSENTHAL, D., GIULI, T., MANIATIS, P., AND MOGUL, J. 2 P2P or Not 2 P2P? *Arxiv preprint cs.NI/0311017* (2003).

[79] ROWSTRON, A., AND DRUSCHEL, P. Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems. In *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)* (Nov. 2001), pp. 329–350.

[80] SETI@HOME. SETI@home Home Page, June 2007. http://setiathome.berkeley.edu/.

[81] SHANNON, C. E. A mathematical theory of communication. *SIGMOBILE Mob. Comput. Commun. Rev. 5*, 1 (2001), 3–55.

[82] SHOKROLLAHI, A. Raptor codes. *IEEE/ACM Transactions on Networking (TON) 14* (2006), 2551–2567.

[83] SIEWIOREK, D. P., AND SWARZ, R. S. *Reliable Computer Systems: Design and Evaluation*, third ed. A. K. Peters, 1998.

[84] SIMITCI, H. *Storage Network Performance Analysis*. Wiley Publishing Inc., 2003.

[85] SINGH, A., AND HAAHR, M. A Survey of P2P Middlewares, June 2007. https://www.cs.tcd.ie/publications/tech-reports/reports.07/TCD-CS-2007-28.pdf.

[86] STOICA, I., MORRIS, R., KARGER, D. R., KAASHOEK, M. F., AND BALAKRISHNAN, H. Chord: A scalable peer-to-peer lookup service for internet applications. In *SIGCOMM* (2001), pp. 149–160.

[87] STRIBLING, J. All-Pairs-Pings for PlanetLab, 2005.

[88] TALIA, D., AND TRUNFLO, P. Toward a synergy between P2P and grids. *Internet Computing, IEEE 7*, 4 (2003), 96.

[89] TAYLOR, I. J. *From P2P to Web Services and Grids: Peers in a Client/Server World*. Springer, 2004.

[90] W. BRAUNEIS, H. L. A New Class of Erasure Codes and its Application to Scalable Multicast Content Delivery. In *Scientific Computing in Salzburg* (2005), vol. 169, Austrian Computer Society.

[91] WEATHERSPOON, H., AND KUBIATOWICZ, J. Erasure coding vs. replication: A quantitative comparison. In *IPTPS '01: Revised Papers from the First International Workshop on Peer-to-Peer Systems* (London, UK, 2002), Springer-Verlag, pp. 328–338.

[92] YIANILOS, P., AND SOBTI, S. The Evolving Field of Distributed Storage. *IEEE Internet Computing 5*, 5 (2001), 35–39.

[93] ZHAO, B. Y., KUBIATOWICZ, J. D., AND JOSEPH, A. D. Tapestry: An Infrastructure for Fault-tolerant Wide-area Location and Routing. Tech. rep., Berkeley, CA, USA, 2001.