Evaluación de la confiabilidad de índices P2P en presencia de alta transitoriedad

Idónea Comunicación de Resultados

ADÁN GEOVANNI MEDRANO CHÁVEZ

Departamento de Ingeniería Eléctrica
Posgrado en Ciencias y Tecnologías de la Información
Universidad Autónoma Metropolitana - Iztapalapa

5 de octubre de 2011

Evaluación de la confiabilidad de índices P2P en presencia de alta transitoriedad

Idónea Comunicación de Resultados para obtener el grado de
MAESTRO EN CIENCIAS Y TECNOLOGÍAS DE LA INFORMACIÓN
2011

Dirigida por la Doctora
Elizabeth Pérez Cortés

Departamento de Ingeniería Eléctrica
Posgrado en Ciencias y Tecnologías de la Información
Universidad Autónoma Metropolitana - Iztapalapa

5 de octubre de 2011

Resumen

Los sistemas par a par (P2P) ofrecen varios servicios para que los usuarios puedan acceder y compartir diferentes tipos de recursos, pares o elementos disponibles, de manera distribuida. No obstante, en estos sistemas ocurre un fenómeno al que llamamos *transitoriedad*. Éste es causado por la *entrada, interacción* y *salida* de un porcentaje considerable de pares y provoca aleatoriedad tanto en el cardinal de la membresía como en la composición de ésta.

Para descubrir los recursos compartidos por los pares, los sistemas P2P hacen uso de un servicio de localización que puede emplear estructuras de datos distribuidas, llamadas *índices P2P*.

La transitoriedad afecta a los índices P2P ya que éstos son una colección de referencias (apuntadores). Su correcto funcionamiento, es decir su *confiabilidad*, depende de que las referencias a los recursos permanezcan sin errores.

Para hacer frente a la transitoriedad, los índices emplean mecanismos con parámetros ajustables cuya función es detectar referencias erróneas y corregirlas. Además, las referencias se almacenan y organizan de tal forma que el índice se puede mantener correcto a pesar de experimentar cierta cantidad de entradas y salidas.

En esta investigación definimos un marco de evaluación de la confiabilidad con el que analizamos a los índices Chord, Pastry y Kademlia en un ambiente de alta transitoriedad en el que la membresía sufre desde $0.125\,\%$ hasta $2\,\%$ cambios por segundo. De los resultados que obtuvimos identificamos qué mecanismos ayudan a los índices a ser más resistentes a la transitoriedad y pueden contribuir al diseño de nuevas estructuras distribuidas.

Existen varios héroes,
pero ninguno como aquel
que peleó con tanto amor
por mi familia y
por sus convicciones.
En memoria a mi padre.

A la inspiración de mi héroe,
y quien yo quiero
de todo corazón.
A mi madre.

Agradecimientos

Evita el fin del mundo, con el brillo de tu armadura y
tus sueños también . . .

Chikyuugi

Quiero agradecer a mi asesora, la Dra. Elizabeth Pérez Cortés por haberme guiado desde mi ingreso al Posgrado en Ciencias y Tecnologías de la Información y por haberme dado la oportunidad de desarrollarme intelectualmente bajo su instrucción. Quiero resaltar la gran admiración y el respeto que le tengo a ella.

También es importante reconocer a los sinodales que dedicaron su tiempo y empeño en orientar este trabajo con sus crítica. En especial, al Dr. Miguel López Guerrero, que me auxilió en la resolución de dudas que surgieron a lo largo de esta investigación.

Mi familia fue esencial para motivar esta causa y seguir adelante. Son el pilar que siempre está ahí, sosteniéndome, y siempre los tengo presentes, en singular a ti mamá.

Mis hermanos también merecen mención, por escucharme en mis momentos difíciles y ayudarme incondicionalmente en mi trabajo, en especial, Raziel Carvajal y Ernesto Galindez.

El presente tiene la esencia de cada uno de ellos, muchas gracias a todos.

Índice

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

Los sistemas par a par (*P2P systems*) son una alternativa al modelo clásico cliente-servidor en la que la funcionalidad del sistema se reparte equitativamente entre los nodos que lo componen para evitar el uso de servicios centralizados. Los usuarios comparten sus recursos de manera distribuida y pueden actuar como clientes o servidores. Principalmente, la tecnología P2P se utiliza para transmitir música y video *al vuelo*, realizar video conferencias y hacer llamadas a través del protocolo VoIP [Or11].

En los sistemas P2P, para que los usuarios puedan encontrar los recursos que otros comparten, se hace uso de un *servicio de localización* que puede funcionar mediante diferentes mecanismos de búsqueda. Es conveniente que este servicio sea rápido y escalable, es decir, que el sistema pueda proporcionar la información para localizar algún recurso de forma eficiente, independientemente del tamaño del sistema.

Durante el desarrollo de la tecnología P2P se han propuesto e implementado una amplia variedad de mecanismos de búsqueda, como la inundación y el indexado distribuido [Me08]; a las diferentes variantes del último las llamamos *índices P2P*.

Generalmente, los índices P2P se presentan en forma de tabla de dispersión distribuida (*distributed hash table*). Estas tablas son un simple conjunto de referencias que permite almacenar y recuperar los recursos [Da03]. Además, estas estructuras distribuidas poseen el atributo de escalabilidad; las operaciones que realizan (buscar, guardar y recuperar), generalmente requieren almacenamiento e intercambio de mensajes con complejidad computacional logarítmica.

Problema

De manera natural, los sistemas P2P forman una red superpuesta integrada con miles o millones de nodos que contribuyen con sus recursos. Estos nodos poseen una propiedad importante: no se puede asegurar su presencia en un instante dado. En consecuencia, los índices P2P deben ser diseñados para operar en un sistema que sufre variaciones abruptas en su membresía. A este fenómeno le llamamos transitoriedad (*churn*).

Para poder hacer frente al dinamismo descrito anteriormente, en los índices P2P se organizan y almacenan las referencias de tal forma que éstos pueden soportar y sobreponerse a cierto nivel de transitoriedad en el sistema. Además, los índices P2P emplean mecanismos para mantener correctas las referencias a los recursos. Sin embargo, si el nivel de transitoriedad es alto, los mecanismos pueden no ser suficientes, ocasionando que las búsquedas por recursos existentes fallen y, en el peor de los casos, el índice pierda completamente su funcionalidad. Por tales motivos, es importante evaluar qué tan confiable es un índice P2P, es decir, si pueden realizar sus funciones correctamente, aun en presencia de *alta transitoriedad*.

Objetivos

En esta investigación analizamos qué tan confiable es un índice respecto a la capacidad que tiene para hacer sus tareas en presencia de alta transitoriedad. Para atender el problema presentado planteamos los siguiente objetivos.

Objetivo general: Analizar la confiabilidad de los índices P2P en presencia de alta transitoriedad.

Objetivos particulares:

1. Comprender algunas de las propuestas de índices P2P y los mecanismos que utilizan para mantener la coherencia ante la transitoriedad.
2. Conocer las medidas existentes para cuantificar la confiabilidad y transitoriedad.
3. Definir un marco de evaluación de confiabilidad para índices P2P.
4. Evaluar la confiabilidad de los índices P2P.

Metodología

A continuación, se describe la metodología adoptada para alcanzar los objetivos antes mencionados:

1. Investigación de las características generales de los sistemas P2P.
2. Exploración del funcionamiento de índices P2P.
3. Identificación, definición y validación de medidas sobre evaluación de índices P2P (estado del arte).
4. Selección de la plataforma de simulación.
5. Especificación de un marco de evaluación para índices P2P.
6. Selección y evaluación de índices P2P.

Aportes

Realizamos un estado del arte sobre evaluación de la confiabilidad de índices P2P. También, determinamos una medida para cuantificar la transitoriedad. Aunado a lo anterior, definimos un marco de evaluación de confiabilidad de índices P2P con el que analizamos a los índices Chord, Pastry y Kademlia. Identificamos qué mecanismos ayudan a los índices a ser más resistentes a la transitoriedad y pueden contribuir al diseño de nuevas estructuras distribuidas.

Estructura del documento

Este documento está organizado de la siguiente manera. En el capítulo 2 se muestran conocimientos básicos sobre sistemas P2P. Posteriormente, en el capítulo 3 se explica el funcionamiento y las características de tres índices P2P. En el capítulo 4, se presenta el estado del arte sobre el cómo la comunidad interesada en índices P2P evalúa su confiabilidad. El desarrollo, marco de evaluación de confiabilidad de índices P2P, y resultados obtenidos se dan a conocer en el capítulo 5. Por último, se exponen las conclusiones y trabajo futuro en el capítulo 6.

Capítulo 2

Sistemas P2P

En este capítulo presentamos las definiciones relevantes para este trabajo. En principio explicamos qué son los sistemas P2P y cómo se clasifican por su arquitectura. Posteriormente, definimos el servicio de localización y sus cualidades en redes estructuradas. Por último, tratamos el problema de la transitoriedad y definimos la característica de interés en esta investigación: la confiabilidad.

2.1. Definición de sistemas P2P

Los sistemas par a par (P2P) surgen como una alternativa en arquitectura de los sistemas distribuidos. Se definen como redes *dinámicas* que tienen la finalidad de compartir recursos de forma distribuida. Estos recursos pueden ir desde archivos multimedia, datos, memoria hasta ciclos de procesamiento. La red está integrada por nodos llamados *pares* [Ste06] que poseen características especiales:

- *Autonomía*: Los pares no siguen regla alguna sobre la manera con la que deben actuar con el sistema ya que, en principio, el sistema no impone restricciones sobre cuántos y de qué forma se deben compartir los recursos. La autonomía de los pares les permite decidir su tiempo de permanencia en el sistema, ésto implica que la composición y cantidad de los pares es variable.
- *Simetría*: Los pares pueden actuar como clientes, solicitando y haciendo uso de los recursos que otros pares comparten; o servidores, compartiendo los recursos que proporcionan al sistema.

Debido a los puntos anteriores, los sistemas P2P obtienen características para organizar a los pares y respetar las propiedades de éstos. Las características [Ste06] que se enuncian a continuación son generales y es probable que un sistema P2P específico no las posea todas:

- *Descentralización*: Los recursos de interés se encuentran distribuidos entre los pares, por tal motivo se usan pocos o ningún servicio centralizado. Si se hace uso de alguna entidad centralizada, ésta solo es empleada como un servidor que gestiona las consultas de los pares, sin embargo, el intercambio de recursos se hace directamente entre los pares.
- *Autoorganización*: El sistema realiza de forma automática las interconexiones posibles para integrar a los pares a la red y darles el acceso a los recursos.
- *Balance de carga*: El flujo de datos se reparte naturalmente entre los pares que se encuentran en el sistema ya que los recursos y la mayor parte de la funcionalidad del sistema residen distribuidamente en ellos.

- *Espacio de nombres*: Es común que los pares se unan al sistema con una dirección IP diferente en cada ocasión, por lo que se hace uso de un espacio de nombres superpuesto al de Internet para localizar tanto a los pares como a los recursos por un identificador y no por su dirección IP.
- *Escalabilidad*: Se superan los problemas de escalabilidad inherentes al modelo cliente-servidor[1] ya que el sistema no depende de una entidad centralizada para otorgar los recursos.

2.2. Clasificación de sistemas P2P por arquitectura

Los sistemas par a par también son redes superpuestas, es decir, se forman estableciendo conexiones lógicas entre los pares sobre la capa de Internet.

Según la arquitectura de la red superpuesta, es decir, la manera con la que se establecen las conexiones entre los pares, los sistemas P2P pueden ser clasificados en estructurados y no-estructurados [Ste06].

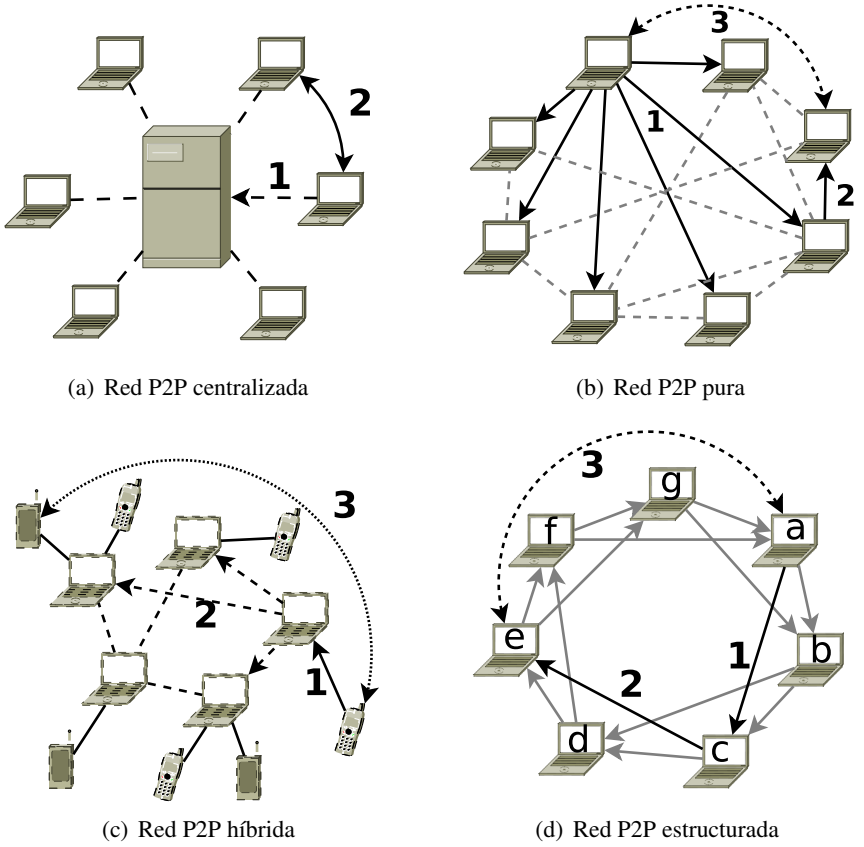
2.2.1. Sistemas P2P no-estructurados

En estos sistemas las interconexiones entre los pares se establecen de acuerdo a reglas flexibles, de manera jerárquica [Lua05] o por medio de un servidor. A continuación se muestra cómo se subclasifican. En las figuras 2.1(a), 2.1(b) y 2.1(c) se puede apreciar un ejemplo gráfico de la topología de estos sistemas:

- *Redes P2P centralizadas*: Fueron las primeras en aparecer, se hacen populares en sistemas como Napster [Yi01] alrededor del año 1998. Emplean un servidor que posee las referencias a los recursos que comparten los pares. Cabe resaltar que el intercambio de recursos se realiza de forma directa entre los pares. La topología típica de estos sistemas es una red estrella. En la figura 2.1(a) se muestra un ejemplo de red P2P centralizada. Las líneas segmentadas representan los enlaces lógicos al servidor, las líneas continuas representan la comunicación entre pares y las flechas representan una consulta. Ésta consiste en que un par solicita un recurso al servidor y éste le responde con una dirección (paso 1) y después se comunica directamente con el par que posee dicho recurso (paso 2).
- *Redes P2P puras*: Estos sistemas no emplean ningún servicio centralizado. Cronológicamente aparecieron a continuación de las redes P2P centralizadas en sistemas como la primera versión de Gnutella [Yi01], del año 2000. Las conexiones entre los pares se establecen cuando un par se comunica con sus vecinos. Los pares vecinos se definen empleando algún criterio de proximidad. La topología de estos sistemas es un grafo aleatorio. La figura 2.1(b) ilustra un ejemplo de este tipo de red. Las líneas segmentadas representan los enlaces lógicos entre pares vecinos y las líneas continuas representan un proceso de comunicación. Las flechas indican algún proceso de intercambio de recursos.
- *Redes P2P híbridas*: Son una combinación de los enfoques anteriores en la que los pares se organizan de manera jerárquica clasificándolos en pares y superpares. Los pares se conectan a los superpares y los superpares se conectan a otros de su misma índole formando una red de cúmulos. Los superpares son entidades centrales dinámicas que actúan como servidores y que pueden ser sustituidas por otros pares si es que fallan o se desconectan. Generalmente, son elegidos por las características de su hardware. Los superpares tienen la función de gestionar las consultas

[1] Modelo clásico en la Internet. Emplea una computadora con hardware potente para poder otorgar diferentes recursos a los clientes que lo acceden.

de los pares que se han conectado a ellos y hacérselas llegar a otros superpares si es necesario. También, actúan como los pares comunes realizando consultas, compartiendo sus recursos o haciendo uso de los recursos de otros pares. Morpheus [Pou05] y la segunda versión de Gnutella [Pou05] son ejemplos de estos sistemas, aparecieron alrededor del año 2002. La figura 2.1(c) representa una red híbrida. Se puede apreciar como los dispositivos con menor hardware (PDA y teléfono móvil) se conectan a los dispositivos con mayor hardware (computadora portátil). Las líneas continuas representan la conexión lógica entre par y superpar, las líneas segmentadas indican la conexión entre superpares y la línea punteada la comunicación entre pares. El paso uno implica la solicitud de un recurso de par a superpar. El paso dos representa la difusión de la consulta entre la red de superpares. En el paso tres, el par que inicio la consulta recibe la dirección del par que tiene el recurso que solicitó y comienza el intercambio de éste.



(a) Red P2P centralizada

(b) Red P2P pura

(c) Red P2P híbrida

(d) Red P2P estructurada

Figura 2.1: Clasificación de Redes P2P según su arquitectura

2.2.2. Sistemas P2P estructurados

En estos sistemas las conexiones entre los pares se establecen siguiendo el criterio de algún índice distribuido que es soportado por todos los pares en el sistema. Definimos índice distribuido de la siguiente manera:

Def. **Índice distribuido P2P.** *Colección de apuntadores que son almacenados de manera simétrica por todos los pares del sistema. Permite organizar y administrar (buscar, dar de alta o dar de baja) los recursos que se comparten en la red P2P a través de indirección.*

Regularmente, los índices son una tabla de dispersión distribuida (*distributed hash table*). En consecuencia, la ubicación de los pares y del contenido es pseudoaleatoria porque los pares emplean algún mecanismo determinista que los coloca en alguna posición fija dentro de un espacio de claves. Cada par almacena una pequeña tabla de encaminamiento que contiene al menos el identificador y la dirección IP de pares que cumplen con alguna medida de proximidad entre sus identificadores [Lua05]. Estos sistemas fueron desarrollados por la comunidad científica y entre algunos de ellos se encuentran Chord [Sto01], Pastry [Ro01], CAN [Ra01], Kademlia [Ma02], Tapestry [Zha01], Kelips [Gu03], Koorde [Ka03] y Viceroy [Mal02]. La topología de estos sistemas puede ser representada como un red superpuesta anillo, árbol, mariposa y otras.

La figura 2.1(d) ejemplifica una red estructurada tipo anillo. Los identificadores de los pares son las letras minúsculas. Las líneas grises representan los apuntadores hacia los dos siguientes pares según el orden del abecedario. Las flechas en negro ilustran un proceso de solicitud donde a pide un recurso de e (pasos uno y dos). Cuando a recibe la dirección de e se comunica directamente con él y comienza el intercambio de recursos descrito por el paso tres y representado por la flecha segmentada.

2.3. El servicio de localización en redes estructuradas

Los sistemas P2P emplean un servicio de localización, el cual definimos de la siguiente manera:

Def. **Servicio de localización.** *Abstracción empleada en los sistemas P2P para localizar, o* descubrir*, los recursos compartidos por los pares [Me08].*

El servicio de localización emplea mecanismos de búsqueda que pueden ser de diferente naturaleza y que pueden usar bases de datos, claves, métodos semánticos y otros. Las características del servicio de localización dependen de cómo se atiendan las siguientes necesidades:

- La manera de formular las consultas.
- La complejidad en memoria y el cómo se organiza el almacenamiento de los recursos.
- La precisión de la búsqueda, entendiendo por precisión la cantidad de recursos relevantes que se obtienen de una búsqueda.
- La rapidez con la que se localizan los objetos.

El tipo de red superpuesta determina el mecanismo de localización que se emplea. En las redes estructuradas se utilizan índices distribuidos, generalmente tablas de dispersión distribuida (*distributed hash table*) definidas a continuación.

Def. **Tabla de dispersión distribuida.** *Estructura de datos distribuida que provee un simple almacén de referencias que permite guardar y recuperar los recursos en un sistema P2P con la misma funcionalidad de una tabla de dispersión común [Da03]. Mapea los recursos en duplas del tipo* $< clave, valor >$*, empleando para ello una función de dispersión, como SHA-1 [SHA], para generar claves que tienen muy alta probabilidad de ser únicas y que sirven para identificar a pares y recursos.*

En seguida mostramos el funcionamiento básico de las operaciones que permiten guardar y recuperar los recursos en las tablas de dispersión distribuida:

- Para guardar un recurso:

1. Se calcula la clave key del recurso que se desea publicar mediante alguna función de dispersión.
2. Se genera la dupla $2tuple =< key, dir >$ con la clave del recurso y la dirección dir del par que lo posee. En este caso, dir puede ser la dirección IP del par más su puerto UDP.
3. Se hace uso de la función poner `put`($2tuple$), para ello:
   a) Se emplea la función de búsqueda dir = `lookup`(key) para encontrar la ubicación en la tabla de dispersión distribuida del par que almacenará la dupla $2tuple$.
   b) Finalmente, se envía la dupla $2tuple$ a la dirección dir del par que regrese la función de búsqueda.

- Para recuperar un recurso:

1. Se calcula la clave key del recurso que se desea obtener mediante alguna función de dispersión.
2. Se hace uso de la función obtener dir = `get`(key), para ello:
   a) Se emplea la función de búsqueda dir = `lookup`(key) para encontrar la dirección del par que almacena la dupla con clave key
3. Finalmente, el usuario descarga el recurso directamente del par con dirección dir.

Como se puede apreciar, la función de búsqueda es una operación relevante en las tablas de dispersión distribuida ya que es el núcleo de las operaciones guardar y recuperar [Me08] que son de más alto nivel de abstracción.

A lo largo de esta investigación, a la función de búsqueda la llamaremos *protocolo de encaminamiento*, ya que se emplea para dirigir mensajes entre los pares que forman el índice P2P.

Algunas características de las tablas de dispersión distribuidas se presentan en seguida [Ste06]:

- Los pares y los recursos comparten un mismo espacio de claves y cada par almacena un pequeño conjunto de referencias (apuntadores) a otros pares, generalmente del orden de $O(\log N)$, donde N es el número de pares en el sistema. Estas referencias se conocen como *estado del par*.
- Las consultas son dirigidas a través de un pequeño número de pares y debido a la organización de los apuntadores, se puede localizar un recurso con una complejidad, en la mayoría de los casos, de $O(\log N)$ saltos.
- El servicio de búsqueda basado en claves tiene la particularidad de encontrar de forma exacta el recurso deseado, por lo que todos los recursos tiene la misma probabilidad de ser descubiertos.
- Las consultas deben ser exactas debido al uso de funciones de dispersión para generar las claves, por ello solo se obtienen referencias a un único recurso. Por ejemplo, al aplicar la función de dispersión al nombre de un archivo, como *poema.txt*, se obtiene cierta clave que con muy alta probabilidad será diferente a la clave de *poem.txt*, por lo que las referencias a los dos recursos serán diferentes aunque sean el mismo archivo. La aplicación puede hacerse cargo de estandarizar los nombres.

2.4. Transitoriedad y confiabilidad

Como se definió al principio de este capítulo, un sistema P2P es una red dinámica en la que ocurre un fenómeno llamado **transitoriedad** o *churn* en inglés, el cual se define de la siguiente manera.

Def. **Transitoriedad.** *Participación activa independiente consistente de la unión, interacción y salida del sistema [Stu06], de un porcentaje considerable de pares en la red [Zhi06].*

La transitoriedad es provocada por varias causas, en principio, los sistemas P2P no establecen un mecanismo que regule el comportamiento de los pares, lo que le otorga a éstos su calidad de autónomos. La autonomía de los pares permite que éstos decidan la manera de interactuar con el sistema, provocando aleatoriedad en la membresía. A continuación, se presenta un análisis de las acciones que toman los pares [Bi07]:

- La unión de los pares implica que éstos inicien una sesión en el sistema. Dependiendo de la red superpuesta y el servicio de localización que se emplee, esta acción implica dar de alta al par y los recursos que comparte, lo que se traduce en anunciar al sistema la existencia de los nuevos pares.
- La participación de los pares es el tiempo de interacción entre el par y el sistema. Durante este lapso, el par hace uso de los recursos de otros pares y a su vez pone a disposición sus propios recursos. La duración de este tiempo es aleatoria y puede depender del tipo de recursos que se comparte, el tiempo promedio de descarga y los hábitos de interacción los usuarios. A la participación también se le conoce como *tiempo de sesión*.
- La salida de los pares corresponde a la acción de abandonar el sistema. Las salidas pueden ser informadas o no informadas. Una salida informada consiste en que el par notifique al sistema su retiro; en la salida no informada el par simplemente abandona el sistema. Bajo el contexto de los sistemas P2P, las fallas son consideradas como salidas no informadas.

Particularmente, el efecto de la transitoriedad es mayor en los índices P2P que en los sistemas no estructurados. Estos últimos, en particular su versión pura, se adaptan mejor al fenómeno debido a la flexibilidad que les otorga su arquitectura libre [Me08]. En cambio, los índices P2P deben emplear mecanismos extras para mantener su coherencia debido a que establecen sus enlaces de manera determinista. Sin embargo, los índices son diseñados de tal manera que pueden resistir cierto nivel de transitoriedad.

El impacto de las entradas y salidas sobre los índices P2P es diferente. La entrada de los pares no es tan perjudicial ya que provoca búsquedas inconsistentes que podrán ser resueltas después de que el sistema anuncie la entrada de los nuevos pares. La salida es más dañina, en especial cuando ocurre de manera no informada, ya que ocasiona rupturas de los enlaces lógicos de la red estructurada. Las búsquedas fallan y en el peor de los casos éstas no se podrán resolver.

Definimos como *protocolo de autoorganización* a las reglas de comunicación que atienden las entradas, salidas informadas y salidas no informadas en un índice P2P. El protocolo de autoorganización, propio de cada índice, hace uso de mecanismos para mantener en coherencia a las conexiones determinadas por los apuntadores entre pares. Estos mecanismos son ajustables y si no se les asigna los valores adecuados a la transitoriedad promedio que experimenta el sistema, los índices P2P pueden perder completamente su funcionalidad.

El fenómeno de la transitoriedad provoca inconsistencia en el servicio de localización, lo que conlleva a búsquedas fallidas en el sentido que es probable que un recurso este disponible y, sin embargo, no se pueda acceder a éste debido a que la referencia se perdió. Diremos que un índice es confiable si su función de búsqueda encuentra un recurso que está dado de alta en un sistema P2P [Cas04] que experimenta transitoriedad. A continuación presentamos nuestra definición.

Def. **Confiabilidad.** *Atributo que mide, con base en la razón de búsquedas exitosas, qué tan capaz es un índice P2P para realizar sus funciones, en presencia de transitoriedad en el sistema.*

La confiabilidad de los índices está en función del desempeño de la función de búsqueda. A su vez, la función de búsqueda depende del cómo se organicen las referencias entre los pares para establecer las conexiones lógicas (el estado de los pares) y del protocolo de autoorganización.

En el siguiente capítulo se presenta el funcionamiento de tres índices P2P. La elección de las siguientes propuestas se debe a que sus protocolos de autoorganización, estado de los pares y topología de red estructurada son diferentes. Como se verá más adelante, estás diferencias influyen en la confiabilidad que cada índice puede otorgar.

Capítulo 3

Índices P2P

En este capítulo presentamos el funcionamiento de los índices P2P Chord [Sto01], Pastry [Ro01] y Kademlia [Ma02], importantes atributos de éstos y la manera cómo hacen frente al fenómeno de la transitoriedad. También ejemplificamos algunos casos concernientes al protocolo de encaminamiento y autoorganización de cada índice y el cómo Chord se ve afectado ante la transitoriedad.

Los índices P2P, en forma de tabla de dispersión distribuida, poseen la ventaja de que tanto el almacenamiento de las referencias a los recursos como la obtención de éstos poseen complejidades logarítmicas. En cuanto a su funcionamiento, particularmente nos interesan las siguientes características:

- *Espacio de claves*: La manera cómo se estructura el espacio de claves. Éste dictamina la topología de la red superpuesta.
- *Estado del par*: La cantidad de referencias que cada par almacena para poder alcanzar a otros pares y el cómo se organizan estas referencias.
- *Protocolo de encaminamiento*: Es el método para dirigir las consultas realizadas por los pares, es decir, el funcionamiento del mecanismo de búsqueda.
- *Protocolo de autoorganización*: Es el mecanismo para gestionar las llegadas y salidas (informadas y no informadas) de los pares. Además mantiene el estado de los pares.

3.1. Chord

Índice propuesto por Ion Stoica *et ál.* en el año 2001 [Sto01]. Según los autores, el objetivo principal de Chord es ofrecer una función de búsqueda eficiente para encontrar los recursos del sistema. El espacio de claves es organizado de manera circular y en orden ascendente, por lo que la topología de la red superpuesta es un anillo. La función de búsqueda se encarga de encontrar los recursos siguiendo un mecanismo de búsqueda unidireccional en el que las consultas se encaminan en sentido horario. Para hacer escalable y rápido al mecanismo de búsqueda, se emplean enlaces lógicos que incrementan su longitud en potencias de dos. Para mantener coherente al índice, Chord hace uso de un protocolo de *estabilización* para autoorganizar los enlaces correspondientes a los pares sucesores y un protocolo de *reparación* que mantiene actualizados los enlaces de potencias de dos.

3.1.1. Espacio de claves

Tanto los recursos como los pares comparten el mismo espacio de claves y éstas se obtienen aplicando una función de dispersión criptográfica, como SHA-1[SHA], al recurso a compartir o a la dirección IP del par. La longitud de los identificadores es de m bits y éstos son arreglados en un espacio circular módulo 2^m, formando una red superpuesta tipo anillo. Las claves de los recursos son almacenadas por el par cuyo identificador es mayor o igual que la clave de los recursos en cuestión; a este par se le conoce como *sucesor*. Por ejemplo, en la figura 3.1 se muestra un espacio de claves Chord de tamaño 2^6. Con esta configuración se obtienen identificadores de longitud $m = 6$ bits que están en el rango $[0, 63]$ en notación decimal. Los identificadores de los recursos están representados con cuadros, mientras que los identificadores de los pares están representados por círculos. En este ejemplo, las fechas que salen de los identificadores de los recursos apuntan al par sucesor que las almacena. Se puede apreciar que las claves 62, 0 y 3 son almacenadas por el par con identificador 7, debido a que la aritmética es módulo 2^6.

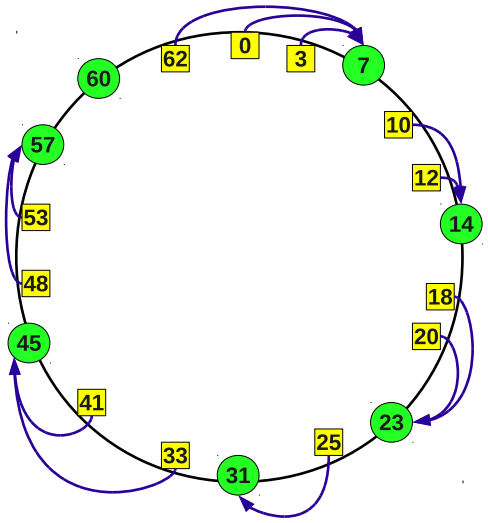


Figura 3.1: Espacio de claves Chord tamaño 2^6.

3.1.2. Estado del par

En Chord, los pares guardan una pequeña cantidad de referencias a otros pares en una tabla de apuntadores (*finger table*) y en una lista de sucesores (*successor list*). El almacenamiento en la tabla de apuntadores es del orden $O(\log N)$, donde N es el número de pares. Por ejemplo, para un espacio de 2^m identificadores cada par almacena m referencias que están compuestas por el identificador del par, su dirección IP y puerto UDP. Para el par n tenemos que la fila i de su tabla contiene el identificador del par sucesor correspondiente a la clave $n + 2^{i-1}$, donde n es el identificador del par en cuestión e $i \in [1, m]$.

Por otro lado, la lista de sucesores[1] contiene típicamente m referencias que apuntan a los m pares sucesores del par n. También se mantiene un apuntador al par predecesor inmediato para facilitar la autoorganización del sistema. En conjunto, el estado de un par Chord tiene una complejidad de $O(2 \cdot \log N)$, donde N es el número de pares. La ilustración 3.2 representa el estado del par 7; éste

[1]La lista al menos almacena la referencia al sucesor inmediato; en un espacio de claves de tamaño 2^m, en [Sto01] se recomienda almacenar m entradas.

posee una tabla de apuntadores de 6 entradas, debido a que el tamaño del espacio de claves es de 2^6, y una lista de sucesores de tamaño seis. Las líneas segmentadas apuntan a los pares de los que tiene conocimiento el par 7 a través de su tabla. Podemos observar que la proximidad entre el identificador del par siete y los identificadores de los pares almacenados en su tabla de apuntadores aumenta en potencias de dos.

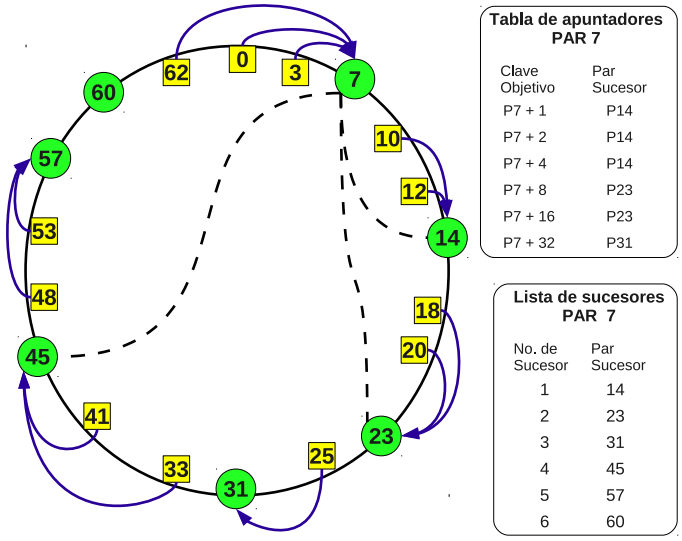


Figura 3.2: Estado del par siete en un espacio de claves Chord tamaño 2^6.

3.1.3. Protocolo de encaminamiento

Debido a la organización del espacio de claves en forma de anillo, cada par Chord está consciente de al menos la dirección de su sucesor inmediato. Las consultas por los recursos son reexpedidas, en orden ascendente con respecto a las claves, por los pares sucesores hasta alcanzar al par que posee la referencia del recurso deseado. Bajo este esquema, se tiene un mecanismo de búsqueda ineficiente con complejidad $O(N)$, donde N es el número de pares participantes.

Para hacer escalable el mecanismo de búsqueda se emplea la información de la tabla de apuntadores. Como se explicó en el apartado correspondiente al estado del par, cada entrada en la tabla de apuntadores almacena la referencia a pares cuya proximidad entre identificadores se incrementa en potencias de dos, por lo que los pares poseen enlaces de gran distancia lógica a otros pares. Cuando un par realiza o recibe una consulta por la clave k de un recurso, éste emplea su tabla de apuntadores y reenvía k al par cuyo identificador sea menor que k y cuya proximidad sea máxima. La consulta se reexpide siguiendo el criterio anterior hasta que se alcanza al par que conoce al par sucesor del recurso k, es decir, el que almacena la referencia al recurso. Por ejemplo, la figura 3.3 ilustra el mecanismo de búsqueda mejorado empleando la tabla de apuntadores. El par 7 realiza una consulta por el par 60, así que envía la consulta al par 45 ya que 45 es menor que 60. Siguiendo el mismo criterio, 45 reenvía la consulta al par 57, ya que 57 es menor que 60 y de entre todas las entradas de la tabla de apuntadores de 45 es la que representa mayor proximidad. En ese punto finaliza la búsqueda ya que 57 sabe que 60 es su sucesor. Las líneas dobles entre los pares 7, 45 y 57 representan el camino tomado por la consulta generada por el par 7.

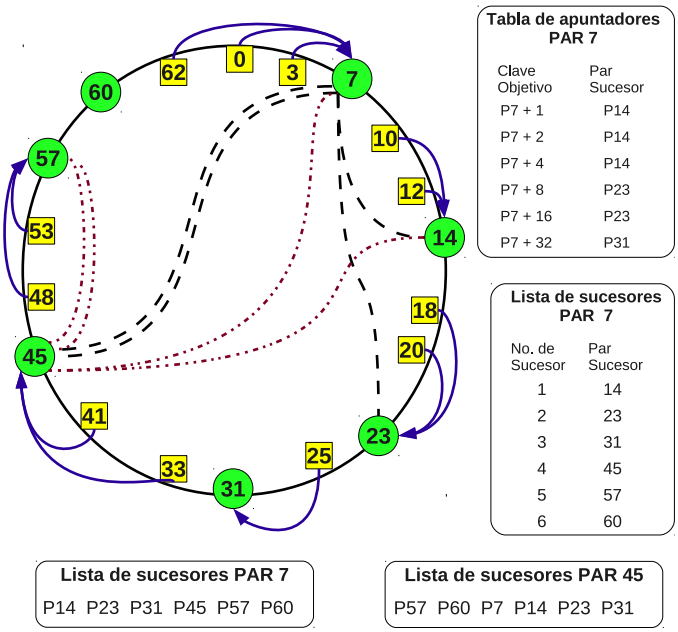


Figura 3.3: Búsqueda empleando lista de apuntadores, el par 7 realiza una consulta para encontrar el par 60.

3.1.4. Autoorganización

Para que las entradas y salidas de los pares tengan el mínimo efecto en la tabla de dispersión distribuida se hace uso de dispersión consistente. La dispersión consistente [Ka97] (*consistent hashing*) es aquella que permite que la adición o retiro de alguna ranura de una tabla de dispersión implique que, en promedio, solo K/r claves sean reasignadas, donde K es el número total de claves y r el número de ranuras, en este caso número de pares. En una tabla de dispersión común, un cambio en su tamaño implica que todas las claves sean reasignadas.

Para que un par n se una al índice Chord, primero debe obtener un identificador, por simplicidad puede elegir uno de manera aleatoria, procurando que el espacio de claves sea lo suficientemente grande para evitar colisiones o, mediante el uso de la función SHA-1 u otra función de dispersión criptográfica. En seguida, mediante algún mecanismo externo[2], el par n realiza una búsqueda preguntando por su propio identificador. El resultado de la búsqueda es la referencia a su par sucesor s. Posteriormente, s actualiza el valor de su apuntador predecesor con n. Después, n comienza a llenar su tabla de apuntadores preguntando a s la identidad de los pares sucesores de las claves $n + 2^{i-1}$, correspondientes a las entradas de la tabla de apuntadores. Hasta este punto, n posee el apuntador a su sucesor inmediato y la tabla de apuntadores, pero los demás pares, a excepción de s, no conocen la presencia de n.

Chord emplea un *protocolo de estabilización* para verificar si el apuntador al sucesor inmediato es correcto. Éste consiste en que un par k pregunta a su sucesor inmediato s la identidad de su par predecesor inmediato p. Si k y p son iguales quiere decir que el apuntador al sucesor inmediato es correcto. En caso contrario, es decir p diferente de k, quiere decir que un nuevo par ha llegado al índice Chord, así que k actualiza su apuntador al sucesor inmediato con la referencia a p y comunica a p que k es su predecesor. El protocolo de estabilización también se emplea para mantener actualizada la lista de sucesores y se ejecuta de manera periódica, pero en caso de ser necesario puede invocarse.

[2]Este mecanismo puede ser un nodo de autoarranque.

Para que los otros pares estén conscientes de la llegada del par n, el predecesor de n debe ejecutar el protocolo de estabilización para que n le dé valor a su apuntador correspondiente al predecesor inmediato p. En ese momento n copia las claves de los recursos que están entre n y p. Por último, el sucesor inmediato s libera las claves de n.

En este punto, el par entrante n puede encaminar consultas provenientes de su predecesor. Solo s y p conocen a n, por lo que las tablas de apuntadores de los otros pares están inconsistentes. Las entradas de las tablas de apuntadores se actualizan mediante un *protocolo de reparación* que pregunta, a menor cadencia que el protocolo de estabilización, la identidad del par correspondiente a la clave $ID + 2^{i-1}$. El protocolo puede ser invocado en caso de ser necesario.

Las salidas informadas se manejan de manera similar a la llegada de un par. Simplemente el par n que abandona el índice le comunica a su sucesor s su salida, entonces n transfiere las claves de los recursos que poseía a s y le hace saber la identidad de su nuevo predecesor inmediato.

Los autores de Chord mencionan que para asegurar que el índice funcione correctamente es necesario que tanto la referencia al par sucesor como las claves de los recursos que almacena cada par sean coherentes. Y para obtener búsquedas rápidas se requiere que la tabla de apuntadores se mantenga actualizada.

En Chord, se hace uso de temporizadores para determinar si un par ha fallado o se retiró del sistema sin informar su salida[3]. Si el par n falla y n está presente en la tabla de apuntadores de otros pares, entonces, los pares que contienen a n deberán buscar al sucesor de n. Sin embargo, la transitoriedad de los pares puede ocasionar que las referencias a los pares contenidos en la tabla de apuntadores estén desactualizadas. Así que, para garantizar búsquedas correctas los pares Chord usan la lista de sucesores ya que ésta se actualiza de forma *ansiosa*[4] y a una cadencia mayor que el protocolo de reparación. En caso de que el sucesor inmediato de n falle, entonces n reemplazará el apuntador por el segundo par en su lista y así. Con esto se obtendrán búsquedas lentas pero correctas y después de cierto tiempo, la tabla de apuntadores se actualizará por medio del protocolo de reparación. Bajo este contexto, el anillo Chord falla solo si todos los nodos en la lista de sucesores fallan.

3.1.5. Ejemplo de transitoriedad

En cualquier índice P2P, las llegadas y salidas, tanto informadas como no informadas, requieren ser atendidas para mantener coherentes los enlaces lógicos de la red superpuesta. La figura 3.4 ilustra el efecto de inconsistencia debido a la transitoriedad en un índice Chord bajo las siguientes condiciones:

1. Los pares 14, 23 y 31 abandonaron el sistema de manera no informada.

2. El tamaño asignado a la lista de sucesores es dos.

3. El par 7 no ha ejecutado ni el protocolo de estabilización ni el reparación.

Cualquier consulta que llegue al par 7 generará una búsqueda fallida. Por ejemplo, si al par 7 le llega una consulta por la clave 48, siguiendo el protocolo de encaminamiento, enviará la consulta al par 45. Después se percatará de la falla e intentará enviar la petición a 23 y a 14 y también fallarán. Posteriormente, el par 7 tratará de usar la tabla de sucesores pero ésta también está inconsistente, por lo tanto la búsqueda fracasará. Las claves 10, 12, 18, 20, 33 y 41 se perderán, sin embargo, la aplicación que use el índice debe hacerse cargo de su gestión.

[3]En los índices P2P, el efecto causado por la falla de un par o una salida no informada puede tratarse por igual, ver la sección 2.4.

[4]Las entradas de la lista de sucesores se actualizan periódicamente.

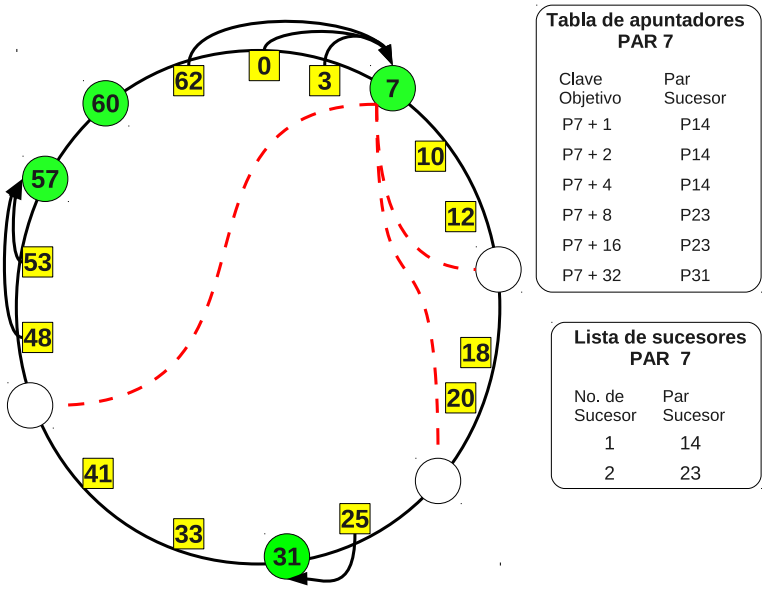


Figura 3.4: Inconsistencia a causa de la transitoriedad en índice Chord, los enlaces del par 7 (líneas segmentadas) eran coherentes antes de la salida de los pares 14, 23 y 31.

3.2. Pastry

Pastry [Ro01] fue propuesto por Rowstron y Druschel en el 2001. Se define como un sistema P2P completamente descentralizado, escalable y autoorganizado que puede usarse como sustrato para localizar objetos en una red superpuesta. Al igual que Chord, el espacio de claves está ordenado de manera circular, pero las claves de los recursos se asignan a los pares más cercanos. La función de búsqueda se implementa mediante encaminamiento Plaxton [Pla97]. La autoorganización del sistema se ejecuta de manera perezosa al descubrir fallas en la resolución de las búsquedas.

3.2.1. Espacio de claves

Pastry usa un espacio circular de claves cuyo tamaño es de 2^l. Los recursos y pares comparten el mismo espacio de claves[5] y éstas tienen una longitud de l bits, la cual es expresada[6] en base 2^b. Los identificadores de los pares se eligen mediante una función de dispersión aplicada a la dirección IP del par o a una llave pública, de tal forma que las claves de los recursos y pares queden uniformemente distribuidas en el espacio de claves. Las claves de los recursos serán almacenadas por los pares cuyo identificador sea numéricamente más cercano a éstas. Una clave k_i es más cercana a otra clave k_j mientras mayor es el número de dígitos que comparten sus prefijos. Por ejemplo, con $b = 4$ y $l = 28$, la clave $k_1 = ABC3491$ es más próxima a la clave $k_2 = ABC3476$ que a la clave $k_3 = AB54420$, ya que el prefijo $ABC34$ que comparte k_1 y k_2 tiene más dígitos en común que el prefijo AB que tienen en común k_1 y k_3.

[5]Las claves de los pares se conocen como identificadores y las claves de los recursos como claves.
[6]Un valor típico de l es 128 bits y de b es 4.

3.2.2. Estado del par

Cada par Pastry almacena referencias a otros pares en un *conjunto de hojas*, un *vecindario* y una *tabla de encaminamiento*, cada una con distintas características que se expresan a continuación.

i	**Tabla de encaminamiento D**			
0	**0**2212102		**2**2301203	**3**1203203
1		1**1**301233	1**2**230203	1**3**021022
2	10**0**31203	10**1**32102		
3	102**0**0230	102**1**1302	102**2**3211	
4	1023**0**322	1023**1**000	1023**2**121	
5	10233**0**01		10233**2**32	
6			1023312**0**	
7				

Conjunto de hojas H			
10233033	10233021	10233120	10233122
10233001	10233000	10233230	10233232

Vecindario V			
00123223	32001212	01211232	11323311
01221232	02312123	22331111	11232321

Figura 3.5: Estado del par 10233102 para un índice Pastry.

Sea D la tabla de encaminamiento de un par Pastry, como ejemplo mostramos la del par 10233102 representada en la figura 3.5. La tabla D está compuesta por i filas, que están en el rango $i \in [0, \log_{2^b}(N) - 1]$, y j columnas que están en el rango $j \in [0, 2^b - 1]$. Tanto para i como para j, b es la base numérica de los identificadores, que para el caso de este ejemplo $b = 4$.

En la fila i se almacenan referencias a pares cuyo identificador posee un prefijo de longitud i. Además, para la fila i se tiene que el dígito $i + 1$ del identificador es igual al valor de la columna j, el resto de los dígitos es diferente. Si no existe un par que cumpla con el prefijo adecuado entonces la entrada se queda vacía. En la figura 3.5, se resalta el dígito correspondiente a la columna j.

La proximidad de los identificadores se incrementa en proporción a i, por ejemplo, usando la figura 3.5, podemos notar que el identificador 10233232, que está en la fila $i = 5$, es más próximo al identificador del par 10233102, que 22301203, contenido en la fila $i = 0$. Cada entrada en la tabla de encaminamiento tiene, además del identificador del par, su dirección IP[7].

Se puede apreciar que la tabla de encaminamiento D sigue el formato *prefijo - columna - resto del identificador*. Por ejemplo, para la entrada contenida en la columna 1 fila 4 se tiene 1023-1-000, así que 1023 es el prefijo común, 1 el número de columna y los demás dígitos son el resto del identificador.

El conjunto de hojas H mantiene pares cuyo identificador es numéricamente cercano y su cardinal se representa con $|H|$. Su tamaño recomendado es de 2^b o 2^{b+1}. En H, la primera mitad de las referencias en el conjunto de hojas $H_{-|H|/2}$ son numéricamente menores al identificador del par; la otra mitad $H_{+|H|/2}$ son numéricamente mayores.

El vecindario V almacena referencias a pares que son cercanos geográficamente[8] y regularmente no se usa para encaminar mensajes. Típicamente es de tamaño 2^b o 2^{b+1} y su cardinal se representa con $|V|$.

[7] En figura 3.5, hacemos abstracción de la dirección IP ya que solo queremos ejemplificar el encaminamiento a través del espacio de claves.

[8] La proximidad geográfica puede obtenerse contando saltos IP, donde un salto es el paso de un enlace a otro en la red.

3.2.3. Protocolo de encaminamiento

Para encaminar una clave k, un par Pastry p hace uso de las referencias contenidas en su tabla de encaminamiento D y conjunto de hojas H. En principio, cuando p recibe una consulta por k, verifica si k está en el rango de su conjunto de hojas; si es así, la consulta se envía al par con identificador numéricamente más cercano. En este caso la búsqueda finalizaría ya que el par contenido en H debe poseer la referencia al recurso con clave k.

Si k se encuentra fuera del rango de H entonces se emplea la tabla de encaminamiento D. En este caso el par p obtiene la longitud del prefijo que tiene en común con k mediante $l = shl(p, k)$. La función $shl(p, k)$ regresa la longitud del prefijo, en dígitos, compartida entre la clave p y k. Después, p envía la consulta al par q contenido en una fila mayor o igual que l y cuya longitud de prefijo en común $l' = shl(q, k)$ sea mayor que l, es decir, $l' > l$.

La figura 3.6 ejemplifica el proceso de encaminamiento en un índice Pastry. El par 10233102, haciendo uso de su tabla de encaminamiento (ver tabla 3.5), genera una consulta por la clave 32211331 y elige al par $D_{0,3} = 31203203$ para dirigir la consulta. Los cuadros representan a las claves de recursos y los círculos a los pares. Las líneas que unen cuadros y círculos indican qué par o pares poseen la referencia a un recurso dado. Las líneas punteadas describen la ruta que tomó la consulta. Los números en negritas representan el prefijo común entre par y clave. Se puede observar como la consulta se reexpide a pares que poseen un prefijo en común con la clave y como, en cada salto, la longitud del prefijo crece.

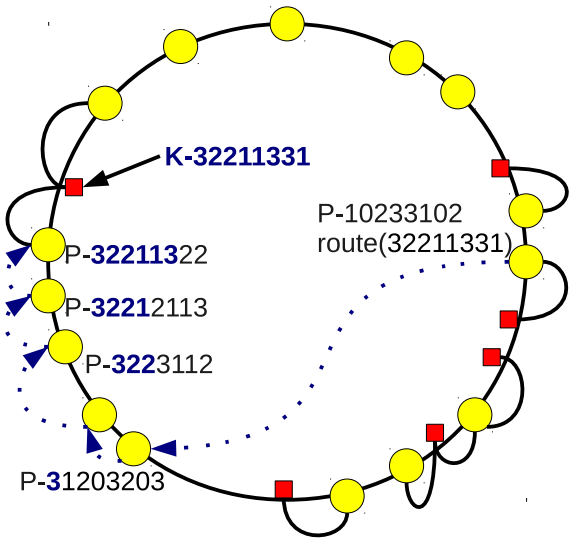


Figura 3.6: Direccionamiento Plaxton en Pastry. Consulta generada por el par 10233102 para hallar la clave 32211331.

3.2.4. Autoorganización

En Pastry, el procedimiento de unión al índice se realiza de la siguiente manera. Primero, un nuevo par obtiene su identificador mediante el uso de una función de dispersión criptográfica, por ejemplo, aplicando la función SHA-1 a la dirección IP o a una clave pública.

Después, haciendo uso de algún mecanismo de autoarranque, un par con identificador P, se comunica con un par con identificador K que se encuentra ya integrado al sistema. Para inicializar su estado, P envía un mensaje de unión a K y K reenvía el mensaje que pasa a través de los pares n_1, n_2,

...hasta n_i, donde n_i es el par numéricamente más cercano a P. En el proceso de unión, P recibirá, como respuesta de su mensaje de unión, información para llenar su estado de la siguiente manera:

- P recibe el vecindario del par K ya que K es próximo a P de acuerdo a una medida de proximidad de red y, bajo el caso de que K y P no compartan un prefijo, también recibe la fila cero de la tabla de encaminamiento D.
- P recibe la fila uno de la tabla de encaminamiento de n_1, ya que, debido al protocolo de encaminamiento, P y n_1 deben tener un dígito en común en su prefijo. Del par n_2, P recibe la fila dos y así sucesivamente hasta completar la fila i por medio del par i.
- Por último, ya que n_i y P son numéricamente próximos, n_i envía su conjunto de hojas a P y P informa su presencia a los pares contenidos en su estado. En este punto el par P se encuentra integrado en el índice Pastry.

La información de encaminamiento enviada a nuevos pares lleva una etiqueta de tiempo. Al momento de completar su estado, el par nuevo regresa su estado a los pares que participaron en el proceso de unión. Éstos verificarán que las etiquetas de tiempo concuerden y en caso de que éstas no empaten, se le solicita al nuevo par que vuelva a comenzar el proceso de unión.

Las salidas informadas y no informadas son tomadas por igual y se detectan perezosamente cuando se encamina una consulta y no se obtiene una respuesta.

Para reemplazar un par contenido en el conjunto de hojas H se envía una consulta al par con mayor índice solicitando su conjunto de hojas H'. Las referencias almacenadas en H' también son válidas para H ya que son cercanas numéricamente y sustituye la entrada fallida por la mejor referencia contenida en H'. Posteriormente se verifica la vivacidad de la referencia seleccionada.

Para reemplazar las entradas ij de la tabla de encaminamiento D de un par P, se solicita a algún par N contenido en la fila i su fila i'. Debido a que i' es una fila válida para D, el par puede copiar la fila entera y verificar la vivacidad de cada referencia. En caso de que la referencia no funcione, se solicita a un par diferente M su fila i''. Con alta probabilidad, este mecanismo asegura que se encontrará un reemplazo adecuado para la entrada ij fallida de D, si es que existe.

Las fallas en el vecindario V se reparan periódicamente verificando la vivacidad de las entradas y con el mismo mecanismo de reparación de H. No se sigue el enfoque perezoso ya que V no se emplea con fines de encaminamiento.

3.3. Kademlia

Kademlia [Ma02] fue propuesto por Maymounkov y Mazières en el 2002. Su espacio de claves es organizado como un árbol binario lleno[9] donde se asignan las claves de los recursos a los pares más próximos empleando la función lógica XOR como medida. El mecanismo de búsqueda sigue un enfoque asíncrono paralelo y además cada que se recibe un mensaje se actualiza el estado del par. Ésto hace que el estado de los pares de Kademlia se mantenga actualizado en medida del intercambio de mensajes.

3.3.1. Espacio de claves

Las referencias a pares y recursos se organizan empleando un árbol binario lleno y su posición en éste se determina por el prefijo más corto. Para ubicar su posición, los pares recorren el árbol en profundidad siguiendo la ruta donde su identificador no está presente.

[9] Un árbol binario lleno es aquel cuyos nodos tienen dos hijos o ninguno.

Las claves de los recursos y pares se obtienen de manera aleatoria, si es que el espacio de claves es lo suficientemente grande para reducir la probabilidad de colisiones, o aplicando una función de dispersión criptográfica. Las claves de los recursos son almacenadas por el par más próximo. La proximidad entre claves está definida por el resultado de aplicar la función lógica XOR a las claves en cuestión. Por ejemplo, la clave $k_1 = 1000$ es almacenada por el par $p_1 = 1011$ en lugar del par $p_2 = 0011$ ya que $p_1\ XOR\ k_1$ tiene un resultado menor que el que da con p_2. La longitud típica de las claves en Kademlia es de 160 bits. La figura 3.7 ilustra un espacio de 2^4 claves e identificadores de 4 bits. Se puede apreciar cómo el par p con identificador 1010 se queda en el tercer nivel del árbol debido a la longitud de su prefijo. Si llegara un par q con identificador 1011, entonces p y q descenderían al cuarto nivel para seguir teniendo un árbol lleno.

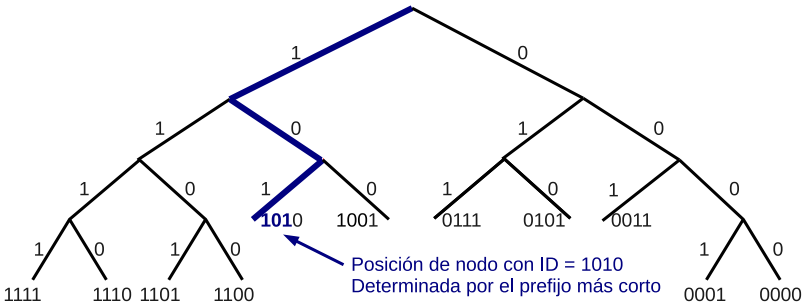


Figura 3.7: Espacio Kademlia con claves de 4 bits.

3.3.2. Estado del par

Los pares Kademlia almacenan referencias a otros pares empleando una especie de lista, que los autores llaman *k-buckets* y son de tamaño k. De manera general, las claves tienen una longitud de l bits, así que por cada bit en el identificador se tiene un *k-bucket*. Por ejemplo, en la figura 3.7 se tiene un espacio de claves de tamaño 16, representadas por cuatro bits. Es por ello que para el par con clave 1100 se tienen cuatro *2-bucket*.

Cada *k-bucket* almacena identificadores con una proximidad específica. Sea i el i-ésimo *k-bucket* correspondiente al bit i de un clave con longitud l. Entonces, tendremos que el i-ésimo *k-bucket* almacena referencias con proximidad entre 2^i y $2^{i+1} - 1$. En la figura 3.7, se tiene que la proximidad entre la clave 1100 y las claves contenidas entre en el *bucket_0* está entre 2^0 y $2^0 - 1$. Para el *bucket_3*, la proximidad entre claves está entre 8 y 15.

También hay que notar que cada *k-bucket* representa un subárbol y que, para cualquier subárbol dado i y pares x y y contenidos en el mismo subárbol, la proximidad entre x y y siempre es mayor que la proximidad entre x y un par z, donde z está contenido en un subárbol distinto. Lo anterior se aprecia en la figura 3.7, ya que las claves contenidas en el *bucket_1* son más próximas entre si, en comparación a la clave almacenada en el *bucket_0*.

Otro punto a considerar es que a medida que la proximidad entre los pares disminuye, es más difícil llenar los *k-buckets*. Kademlia se asegura que cada par conozca al menos un par de cada subárbol, es decir, que al menos exista una referencia a un par en cada *k-bucket*.

3.3.3. Protocolo de encaminamiento

En Kademlia las búsquedas se realizan siguiendo un mecanismo asíncrono paralelo y para ello se emplean las referencias contenidas en los *k-buckets*. Cuando un par P quiere realizar una consulta

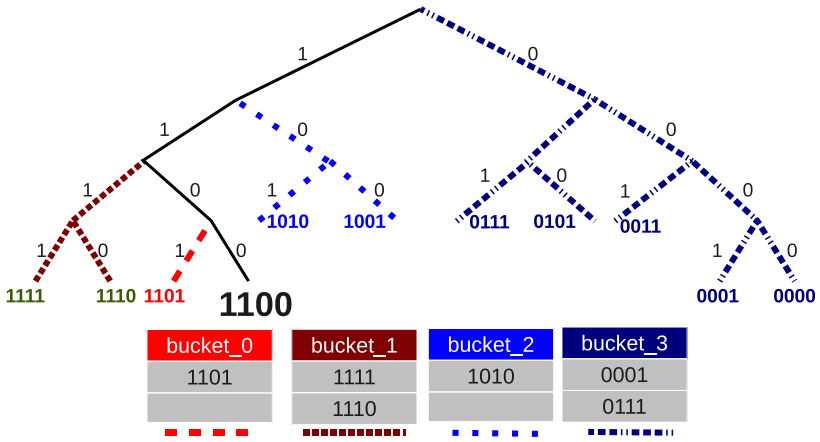


Figura 3.8: Estado del par Kademlia 1100.

por un recurso con clave c, P obtiene α pares próximos con respecto a la medida XOR y les envía la consulta por c. Cuando los α pares reciben la consulta por c, si poseen la referencia a c se la hacen llegar a P, en caso contrario consultarán sus *k-buckets* y regresarán a P la referencia al par más cercano a c que conocen. La consulta se reexpide de manera iterativa y en cada iteración, la proximidad entre las claves se reduce al menos en un medio. En la figura 3.9, el par con identificador 1100 genera una consulta por la clave 0011. Siguiendo el protocolo, 1100 emplea la información de sus *k-buckets* y encuentra que con el par 0001, localizado en el *k-bucket* correspondiente al bit 3, se genera la menor proximidad, así que le envía la consulta. El par 0001, empleando sus *k-buckets* responde con la clave 0011 y la búsqueda finaliza.

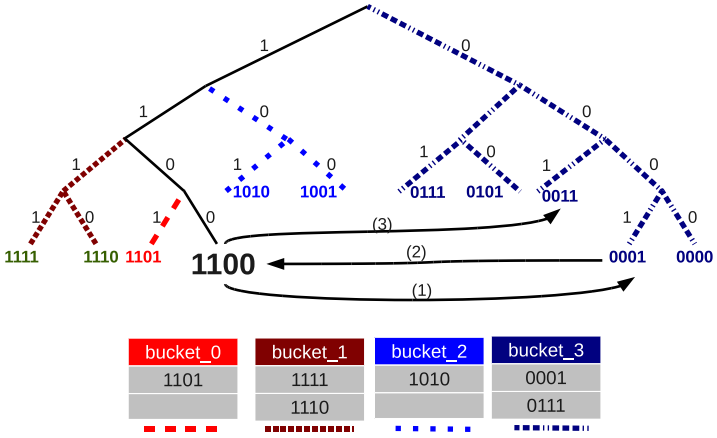


Figura 3.9: Búsqueda con $\alpha = 1$ por la clave 0011, generada por 1100.

3.3.4. Autoorganización

En Kademlia se sigue un enfoque dinámico para la detección de fallas y el mantenimiento del estado, ya que estas acciones dependen del mecanismo de búsqueda y del tráfico de la red. Ésto se debe a que las consultas de Kademlia emplean *piggybacking (acarreo a cuestas)*[10]; al realizar una consulta se adjunta a ésta la información necesaria para actualizar los *k-buckets* de los pares que la

[10]Para mayor entendimiento de la técnica de *piggybacking* (acarreo a cuestas) leer [Me08] apartado 3.3.4.

reciban.

Los *k-buckets* son ordenados según la aparición de los pares, es decir, del último par visto hasta el par visto más recientemente, colocando el último par en la cabeza de la lista y el más reciente en la cola. Cada que un par P recibe un mensaje de otro par Q, P actualiza el *k-bucket* correspondiente a Q y se realiza alguna de las siguientes acciones:

1. Si Q ya estaba en el *k-bucket* de P entonces Q se mueve a la cola, en otro caso:
2. Si aún queda espacio en el *k-bucket* simplemente se agrega la referencia a Q, colocando a éste en la cola, en otro caso:
   a) P verifica la vivacidad del último par visto R, si éste responde entonces R se mueve a la cola del *k-bucket* y se descarta el registro de Q, en caso contrario,
   b) Se remueve R del *k-bucket* y se coloca a Q en la cola.

Como se puede apreciar, los pares vivos nunca son removidos de los *k-bucket*, lo que hace que Kademlia resista ciertos ataques de denegación de servicio y que considere en su diseño a los pares con tiempos de sesión mayor.

Cuando el tráfico disminuye, puede ocurrir que el estado de los pares se torne inconsistente, por lo que los pares Kademlia actualizan sus *k-buckets* cada hora mediante un *protocolo de estabilización*. Este protocolo implica elegir aleatoriamente alguna entrada en los *k-buckets* para comprobar su permanencia, realizando una búsqueda por el identificador del par.

El proceso de unión se realiza de la siguiente forma. En principio, un par P debe conocer la identidad de otro par Q, entonces P inserta a Q en el *k-bucket* apropiado. Después, realiza una búsqueda por su propio identificador. Posteriormente P actualiza los *k-buckets* que están más allá de su vecindario. Por medio de la primera búsqueda y la actualización de los *k-buckets*, P llena su estado y se hace presente en los estados de otros pares, si es que lo requieren.

Las fallas se detectan cuando no se obtiene respuesta de alguno de los pares contenidos en los *k-buckets*. Cuando esto ocurre se remueve la entrada y se emplea otra. Siempre que el tráfico de mensajes sea considerable, los *k-buckets* se mantendrán actualizados.

3.4. Sumario

En las secciones anteriores se presentó el funcionamiento de tres índices P2P: Chord, Pastry y Kademlia. Todos presentan diferencias importantes en la manera en que organizan el espacio de claves, en el estado de los pares, función de búsqueda y autoorganización. Con base en [Ste06] y [Me08], a continuación presentamos la tabla comparativa 3.1 que muestra la complejidad en memoria, que requiere el estado de cada par, así como la complejidad, en número de mensajes, que necesita el encaminamiento de los mensajes y la llegada y salida de un par.

Tabla 3.1: Contraste de complejidad entre Chord, Kademlia y Pastry

Índice	**encaminamiento**	**estado**	**llegadas**	**salidas**
Chord	$O(\frac{1}{2}\log_2(N))$	$O(2\log_2(N))$	$O(2\log_2^2(N))$	$O(2\log_2^2(N))$
Kademlia	$O(\log_2(N))$	$O(2\log_2(N))$	$O(\log_2(N))$	$O(\log_2(N))$
Pastry	$O(\log_{2^b}(N))$	$O(2^b + b\log_{2^b}(N))$	$O(\log_{2^b}(N))$	$O(\log_b(N))$

En la tabla 3.1, N representa el número de pares en el sistema y b la base en la que se encuentran los identificadores. En Kademlia, la base logarítmica puede ser 2^b ya que los autores mencionan que su índice puede ser modificado para funcionar con identificadores base b.

Con respecto a los mecanismos que poseen los índices estudiados para hacer frente a la transitoriedad presentamos las siguientes observaciones:

- Chord:
  - *Topología*: anillo en el que se encaminan los mensajes de manera unidireccional.
  - *Medida de proximidad*: sucesores, pares cuyo identificador es inmediatamente superior.
  - *Detección de inconsistencias:*
    - Protocolo de estabilización: actualiza de manera ansiosa la lista de sucesores.
    - Protocolo de reparación: actualiza la tabla de apuntadores después de intervalos de tiempo más grandes que los del protocolo de estabilización.
- Pastry:
  - *Topología*: anillo que emplea encaminamiento Plaxton.
  - *Medida de proximidad*: igualdad de prefijo, mientras más dígitos tengan en común dos claves, más próximas son. Al usar el vecindario se emplea la diferencia numérica de los identificadores.
  - *Detección de inconsistencias*: se sigue un enfoque perezoso en el que el estado se repara cuando se detecta una falla en el encaminamiento.
- Kademlia
  - *Topología*: árbol binario lleno que emplea enfoque paralelo asíncrono para enviar consultas.
  - *Medida de proximidad*: función XOR, la cercanía entre dos claves está definida por el resultado de la función lógica.
  - *Detección de inconsistencias:*
    - *Acarreo a cuestas*: los mensajes traen adjuntos información necesaria para actualizar el estado de los pares.
    - *Estabilización*: los $k\text{-}buckets$ se actualizan después intervalos de tiempo de ~ 1 hora.
    - *Consistencia*: en el estado se toma en cuenta los pares que exhiben mayor tiempo de sesión.

Capítulo 4

Evaluación de índices P2P: estado del arte

Este capítulo presentamos algunas de las propuestas realizadas por la comunidad interesada en los índices P2P y que sirven como base de nuestra investigación. Analizamos las propuestas con base en importantes puntos de interés. Al final mostramos la reflexión que condujo a este trabajo.

4.1. Aspectos de interés

Para realizar esta investigación identificamos, de manera general, cómo la comunidad evalúa a los índices P2P. En particular, tomamos como focos de interés los siguientes aspectos: el tipo de experimento, plataforma de experimentación, modelo de transitoriedad, medidas empleadas e índices evaluados. A continuación se describen los aspectos mencionados.

4.1.1. Índices evaluados

Estamos interesados en los resultados de confiabilidad que la comunidad ha obtenido al evaluar distintos índices P2P. Para mayor entendimiento sobre los índices, explicamos el funcionamiento de Chord [Sto01], Kademlia [Ma02] y Pastry [Ro01] en el capítulo 3. Otros índices P2P propuestos son Koorde [Ka03], Kelips [Gu03], Tapestry [Zha01], CAN [Ra01], Viceroy [Mal02] y Bamboo [Rh04].

4.1.2. Tipos de experimentos

En [Je09] se mencionan cuatro maneras de validar los experimentos: *in situ*, comparativa (*benchmarking*), emulación y simulación, las cuales describimos a continuación..

- *In situ* se refiere a experimentar sobre la aplicación real, por lo tanto se tiene muy poco nivel de abstracción.
- En una *comparativa* se emplea un modelo de una aplicación sobre un sistema real.
- En las *emulaciones* se ejecuta la aplicación por medio de máquinas virtuales o ejecutando directamente el experimento sobre algún tipo de hardware, que no necesariamente es el que se emplea de manera real.

- La *simulación* se enfoca únicamente en una parte dada del ambiente y hace abstracción del resto del sistema. Por medio de simulaciones se pueden realizar una gran cantidad de experimentos bajo distintas condiciones y altamente reproducibles

4.1.3. Plataformas de experimentación

La plataforma de experimentación es el artilugio sobre el cual se ejecuta algún experimento. Para el caso *in situ* encontramos Cruiser [Stu05-1] que es un rastreador que permite capturar el comportamiento de los pares en sistemas P2P estructurados y no estructurados. También está PlanetLab [Planet] el cual puede ser empleado para observar redes superpuestas.

Con respecto a las emulaciones hallamos ModelNet [Va02]. Ésta es una herramienta que permite evaluar sistemas distribuidos de gran escala como sistemas P2P. Bajo esta herramienta, las aplicaciones experimentan fenómenos comunes presentes en la Internet, como limitaciones en el ancho de banda, latencia y tasa de pérdidas.

En el estado del arte encontramos investigaciones que empleaban los simuladores P2Psim [P2Psim] y NetHawk EAST [NetHawk]. En el anexo A se encuentra una discusión sobre simuladores P2P.

4.1.4. Modelos de transitoriedad

Es el esquema mediante el cual se hace abstracción de las entradas, fallas y salidas del sistema P2P. Pueden presentarse dos maneras:

- *Transitoriedad por proceso de Poisson con tasa de eventos global*: a un sistema P2P se le asigna una tasa global de entradas y salidas mediante un proceso de Poisson con tasa λ. Ambos eventos, entradas y salidas, se modelan con el mismo proceso de Poisson. Las salidas ocurren aleatoriamente después de cierto tiempo distribuido exponencialmente. Las entradas ocurren inmediatamente después de la salida de un par. Bajo este modelo los experimentos no escalan ya que no se obtiene la misma transitoriedad estableciendo la misma tasa con diferentes tamaños de red.

- *Transitoriedad por porcentaje de salidas*: consiste en, dado un número de pares N, provocar que cierto porcentaje de N abandone el sistema. Este modelo escala en los experimentos, pero no considera las llegadas de los pares.

- *Transitoriedad a nivel de nodo con N fija*: Las salidas y llegadas se generan mediante el uso de dos variables aleatorias: T_{ON}, que describe el tiempo de interacción de los pares y T_{OFF}, que describe el tiempo que permanecen los pares fuera del sistema. Los retiros ocurren cuando el tiempo T_{ON} asignado a un par se agota. Las llegadas ocurren cuando el tiempo T_{OFF} asignado a un par se agote. Bajo este modelo se crean un número N de pares total, y cada par conmuta entre los estados ON y OFF. Cuando están en ON participan con el sistema, cuando están en OFF hacen nada. El número de pares que están en ON es N'. Este modelo es escalable ya que el sistema experimenta la misma transitoriedad independientemente de N.

- *Transitoriedad a nivel de nodo con N variable*: Es una variante del anterior, las llegadas al sistema ocurren después de un tiempo descrito por una variable aleatoria T_{dead}, que se dispara cuando un par sale del sistema. Las salidas ocurren cuando la variable aleatoria T_{live}, que describe el tiempo de vida[1] de un par, se agota. Bajo esta abstracción, no se conoce directamente

[1] En este modelo, el tiempo de vida T_{live} es similar al tiempo de sesión T_{ON} de los pares.

el número de pares promedio N' ni el número total de pares N. Los pares no regresan cuando mueren, pero después de la salida de un par se deja pasar un tiempo muerto T_{dead} para que llegue otro par. Este modelo también es escalable, ya que es independiente de N.

4.1.5. Medidas empleadas

Para evaluar diferentes propiedades de los sistemas P2P, es necesario el empleo de medidas que cuantifiquen alguna característica de interés. En particular, nos interesamos en medir la confiabilidad y la transitoriedad. De igual manera, es importante el uso de una medida que describa rendimiento de los índices en términos de consumo ancho de banda.

Ninguna de las medidas de transitoriedad que mostramos a continuación describe completamente el fénomeno de la transitoriedad descrito en la sección 2.4 de esta investigación.

- *Tasa de transitoriedad*: de manera global, describe las entradas y salidas de pares por unidad de tiempo que ocurren en un sistema P2P. Se denota con la tasa λ y sus unidades son $pares/s$. Generalmente, se emplea en el modelo de transitoriedad por proceso de Poisson. Esta medida cuantifica directamente la transitoriedad del sistema, es decir, otorga información del número de salidas y llegadas que ocurren por unidad de tiempo.

- *Porcentaje de abandonos*: Esta medida es usada en la variante del modelo de transitoriedad global y cuantifica la cantidad de pares que abandonan la red. Mide directamente las salidas en el sistema y es escalable. No toma en cuenta las llegadas de los pares.

- *Tiempo de sesión*: Es la duración, en unidad de tiempo, del lapso que permanecen los pares en línea. Esta medida se emplea generalmente en el modelo de transitoriedad a nivel de nodo y regularmente se mide en $segundos$; sin embargo, cuantifica la transitoriedad de forma indirecta ya que no da información sobre el número de llegadas y salidas que experimenta el sistema.

Con respecto a la confiabilidad, y bajo la premisa de que un índice es confiable si su función de búsqueda encuentra un recurso que está dado de alta en el sistema P2P [Cas04], consideramos que las medidas de confiabilidad están descritas por el desempeño de las búsquedas en el sistema. Estas son las medidas encontradas:

- *Razón de búsquedas exitosas*: Es el cociente de las búsquedas exitosas sobre el número total de búsquedas. Se puede expresar en porcentaje.

- *Retardo de búsqueda promedio*: Es el tiempo que tarda en resolverse una búsqueda exitosa, se puede expresar en segundos.

- *Tasa de pérdidas*: Es el número de búsquedas fallidas por unidad de tiempo.

- *Tasa de búsquedas exitosas*: Es el número de búsquedas exitosas por unidad de tiempo.

- *Punto de Quiebra:* Si x porcentaje de pares abandona simultáneamente un sistema P2P, y este abandono ocasiona que el 50 % de las búsquedas, generadas aleatoriamente, fallen, entonces x es el punto de quiebra del sistema.

El que un índice otorgue cierta cantidad de confiabilidad depende en gran medida de la manera con que se configure su estado y protocolo de autoorganización. Este protocolo envía mensajes de mantenimiento para que el estado de los pares permanezca sin errores. Por tal motivo, es importante observar cuántos recursos se consumen para adquirir cierta cantidad de confiabilidad. Las siguientes medidas evalúan el consumo de recursos en términos del ancho de banda.

- *Ancho de banda promedio*: Es el número de bytes promedio que envía un par por unidad de tiempo. Es expresada en $bytes/par/s$. Para simplificar la notación, en esta investigación usaremos Bps/par, donde Bps son bytes por segundo.

- *Número de mensajes promedio*: Una abstracción de la anterior, es la cantidad de mensajes que un par envía por unidad de tiempo. Estos mensajes pueden involucrar en general cualquier mensaje o solo los que implican mantenimiento, sus unidades son $mensajes/par/s$. De igual manera, para evitar ambigüedades en la notación usaremos mps/par, donde mps son mensajes por segundo.

4.2. Trabajos relacionados

En esta sección, y respaldándonos en los puntos planteados en la sección anterior, se presentan algunos aportes de la comunidad necesarios para poder evaluar la confiabilidad en los índices P2P y que sirven como base para nuestra investigación.

Jinyang Li *et ál.* [Li04] presentan una evaluación que busca encontrar el equilibrio entre el costo de mantener consistencia en el estado de los pares y el desempeño de la búsqueda de los índices P2P Tapestry, Chord, Kelips y Kademlia. Los experimentos los realizan mediante simulación empleando como plataforma P2Psim [P2Psim]. Estos experimentos consisten en someter a transitoriedad, empleando el modelo de transitoriedad a nivel de nodo con N fija, a 1024 pares que tenían latencias derivadas de medir los retardos por pareja de 1024 servidores DNS[2] usando el método King [Kr02]. Las medidas que emplearon fueron ancho de banda promedio y retardo de búsqueda promedio[3]. Los autores encontraron que los índices analizados pueden ser configurados para alcanzar latencias menores a 250 ms. También analizaron por separado el desempeño del ajuste de un único parámetro por índice. Encontraron que los desempeños son similares, siempre y cuando, los parámetros del mecanismo de autoorganización, propios de cada índice, estén correctamente calibrados. Concluyen con que asignar los valores se convierte en una tarea compleja, ya que, parámetros similares en diferentes índices ofrecen comportamientos diferentes.

Sean Rhea *et ál.* [Rh04] mencionan que los índices P2P no pueden manejar altas tasas de transitoriedad, como lo hacen los sistemas no estructurados y, mediante el análisis de las deficiencias de los sistemas estructurados, proponen el diseño de un índice llamado Bamboo[4], que está basado en Pastry. Los autores, mediante emulación y empleando ModelNet como plataforma, evaluaron el desempeño de Chord, Pastry y su propuesta Bamboo. La transitoriedad la modelaron mediante un proceso de Poisson; cada que un par muere, otro se integra manteniendo así un tamaño de red constante. Las tasas promedio de entradas/salidas que emplearon van desde 1.4 minutos a 3 horas. Las medidas que utilizaron para evaluar el desempeño de los índices fueron la razón de búsquedas exitosas (expresada en porcentaje), el retardo promedio y ancho de banda promedio (expresado en Bps/par). Para medir la transitoriedad usan el tiempo de sesión. En Pastry encontraron que la eficiencia de las búsquedas exitosas disminuye en gran medida con una tasa alta de transitoriedad. También encontraron que las búsquedas exitosas aumentan cuando se le da al índice tiempo para estabilizarse, sin embargo, los autores mencionan que en una red real es muy difícil hacer eso. Además observaron que Bamboo presentaba problemas similares a los de Pastry pero, empleando un mecanismo como el de Chord para actualizar el estado, encontraron que se mejora el desempeño de las búsquedas. Para Chord encontraron que, a diferencia de Pastry, la razón de búsquedas exitosas es del 100 %. El problema que éste

[2] *Domain Name Server*

[3] Medida que cuantifica el tiempo que tarda en resolverse una consulta.

[4] Para entender el funcionamiento del índice Bamboo puede dirigirse a la referencia [Rh04].

presenta es que la resolución de la búsqueda tiene una gran latencia. Se menciona que el encaminamiento iterativo puede ser la causa del problema. Su propuesta Bamboo presenta latencias menores que Chord a pesar de experimentar altas tasas de transitoriedad.

Zhiyu Liu *et ál.* [Zhi06] presentan un análisis de confiabilidad de cinco índices: Chord, Tapestry, Kelips, Kademlia y Koorde. El objetivo del trabajo es observar la razón de búsquedas exitosas que los índices pueden otorgar ante lo que ellos llaman *alta transitoriedad*[5]. Los autores mencionan que la razón de búsquedas exitosas está relacionada con la conectividad de la red y por ello definen la medida *punto de quiebra*. Ésta describe indirectamente la fragmentación de un índice ya que cuando un par se aísla o la red se divide entonces se producirán búsquedas fallidas. Para soportar sus hipótesis, los autores realizaron simulaciones empleando P2Psim [P2Psim]. El escenario que emplearon consistía en una red de 1000 pares con tiempo de ida y vuelta (*round trip time*) de 2 segundos. Cada par generaba una búsqueda de manera aleatoria, distribuida exponencialmente con media de 10 segundos. Emplearon el modelo de transitoriedad por porcentaje de salidas, cuya medida es el porcentaje de salidas por unidad de tiempo. En la simulación concerniente a Tapestry, encontraron que cuando el 50 % de los pares fallan, el índice no se recupera a pesar de darle tiempo para estabilizarse. Si la falla es menor al 40 %, el índice se recupera. En seguida se usa el punto de quiebra para comparar índices. Hallaron que el punto de quiebra de Kelips ocurre cuando se provoca que el 70 % de los pares abandonen el índice, mientras que el punto de quiebra de Chord ocurre alrededor del 80 %. Tapestry, Koorde y Kademlia tienen un punto de quiebra similar, en alrededor del 50 % de abandonos, sin embargo, mencionan que Kademlia puede mejorar en redes con mayor número de pares. Por último, y con base a sus resultados, proponen un algoritmo llamado DARE que tiene por objetivo unir las subredes generadas por alta transitoriedad.

Daniel Stutzbach *et ál.* [Stu06] presentan un estudio para caracterizar la transitoriedad de los sistemas P2P. En éste se examina el comportamiento de los pares en Gnutella (sistema no estructurado), BitTorrent (sistema de distribución de contenido) y KAD (implementación del índice Kademlia). El estudio abarca los posibles errores comunes al capturar la transitoriedad, un análisis del comportamiento a nivel de grupo y otro a nivel de par. Para Gnutella, capturaron 5 muestras mediante un rastreador (*crawler*) llamado Cruiser[6]. Las muestras que consiguieron consistían de cinco tomas de estado obtenidas continuamente por períodos de 48 horas, con duraciones de rastreo que iban de 4 a 10 minutos. También utilizaron Cruiser para capturar un subconjunto, o zona, del espacio de identificadores de KAD. Estas zonas están uniformemente distribuidas ya que el mismo espacio de identificadores está distribuido de esa manera y, por lo tanto, una zona es representativa. Las muestras que colectaron fueron capturadas una tras otra, por espacios de 48 horas y de cuatro zonas diferentes. En los sistemas del tipo BitTorrent se emplea un servidor, llamado *tracker*, al que los pares se conectan. A este servidor le son enviados periódicamente algunos datos como el progreso de la descarga de archivos y los tiempos de llegada y salida de los pares, mismos que son registrados en bitácoras con una granularidad de un segundo. En las bitácoras solo se registran las salidas informadas. Las muestras que analizaron provenían de tres redes P2P: *Debian iso images*, *Red Hat iso image* y *Flat Out*. Algunos de sus resultados se describen a continuación:

- La duración de las sesiones de los pares van del orden de minutos hasta días, quizá semanas, y éstas se ajustan a la distribución de Weibull o la log-normal.
- Las inter-llegas se modelan mejor con la distribución Weibull con parámetro de forma $k < 1$.
- La mayoría de los pares permanecen un tiempo considerable en el sistema, y otra menor cantidad

[5] Zhiyu Liu *et ál.* mencionan que la alta transitoriedad ocurre cuando un gran porcentaje de pares se unen, retiran o fallan, de manera frecuente y simultánea, haciendo que el mecanismo de autoorganización no sea suficiente.

[6] Diseñado por Stutzbach *et ál.* [Stu05-1].

de pares se une y abandona el sistema a tasas muy altas. Estos últimos constituyen una gran porción en el número de sesiones.

- El tiempo de actividad de los pares puede ser empleado como un vaticinador del tiempo de actividad restante, pero se debe estar consciente de que éste exhibe una varianza considerable.

Octavio Herrera *et ál.* [He07] proponen un modelo de transitoriedad y sobre éste evalúan el desempeño del encaminamiento y recuperación de contenido de Chord. El modelo se constituye de cuatro componentes que se describe a continuación:

- *Usuario*: clasifica a los usuarios en *benefactores*, *pares* y *mirones*. La diferencia entre éstos radica en el tiempo de sesión, siendo los benefactores los que presentan mayor cantidad de tiempo en el sistema. Las otras dos clases dependen del recurso que se comparte, los pares pueden permanecer de varios minutos a horas. Los mirones se conectan al sistema por poco tiempo.
- *Nodo*: modela, mediante variables aleatorias, tanto el tiempo de actividad como el tiempo de inactividad de un par. Este componente caracteriza la transitoriedad del sistema (transitoriedad a nivel de nodo).
- *Red*: incluye el número de nodos en el sistema y el servicio de localización empleado.
- *Recursos*: son los objetos administrados por la aplicación que hace uso de un servicio de localización.

Bajo este marco los autores evaluaron una implementación estática y una dinámica del índice Chord[7] empleando el simulador P2Psim. Para sus experimentos consideraron el consumo del ancho de banda promedio, el retardo de búsqueda y la tasa de búsquedas exitosas. Con respecto al encaminamiento, los autores encontraron que con los casos extremos de transitoriedad[8] se obtiene una mejor tasa de búsquedas exitosas con Pareto que con la distribución uniforme. Cuando se toma en cuenta la recuperación de contenido se obtiene un peor desempeño, siendo que la implementación estática de Chord obtiene como máximo una tasa de búsquedas exitosas normalizada de 0.8 y en la mayoría de los casos 0.65. La implementación dinámica de Chord muestra que la mitad de los escenarios tienen tasas inferiores a 0.8.

Zhonghong Ou *et ál.* [Zho09] analizan la confiabilidad y desempeño de Kademlia empleando diferentes distribuciones para el modelo de transitoriedad a nivel de nodo. Ellos desarrollaron una implementación de Kademlia que realiza búsquedas seriales[9] y encaminamiento iterativo. Su plataforma de simulación fue NetHawk EAST. La transitoriedad la miden mediante el tiempo de sesión medio; el desempeño a través del ancho de banda medio y el número de mensajes promedio. La confiabilidad la observan mediante la tasa de búsquedas exitosas. Los autores emplearon una red simulada de 400 pares empleando las distribuciones exponencial, Pareto y Weibull para describir los tiempos de sesión de los pares. Se definió que el tiempo de actividad promedio fuese igual que el tiempo de inactividad, es decir, $T_{ON} = T_{OFF}$. Para la distribución Pareto se utilizó $\alpha = 2,3$ y para la Weibull $k = \frac{1}{2}, \frac{1}{3}$. Con respecto al índice, se definió que el tamaño de los *k-buckets* fuese de 3, al igual que el número de búsquedas seriales. Los autores encontraron que las evaluaciones correspondientes al ancho de banda y a la tasa de búsquedas tenían resultados muy similares usando la distribución exponencial y la de Pareto; con Weibull se obtenía un desempeño inferior, en especial cuando el parámetro de forma era $k = 1/3$.

[7] La implementación estática consiste en dejar que los parámetros del protocolo de autoorganización no varíen y la dinámica consiste en que los parámetros se ajusten a la transitoriedad.

[8] Tiempos de sesión mediano de 8 a 120 minutos.

[9] En el índice original Kademlia las búsquedas se hacen de forma paralela asíncrona.

4.3. Evaluación de la confiabilidad

En esta investigación estamos interesados en observar qué tan buenos son los índices P2P para realizar sus tareas ante la presencia del fenómeno de la transitoriedad. Mediante la confiabilidad podemos comparar la calidad relativa de las propuestas existentes haciendo uso de las medidas descritas en la sección 4.1.5.

A partir los aspectos de interés, presentados en 4.1 y del trabajo base presentado en 4.2 observamos los siguientes puntos:

- En [Li04, Zhi06, He07, Zho09] realizan simulaciones para validar sus hipótesis, por lo que consideramos que la simulación es una manera aceptada por la comunidad interesada en índices P2P para realizar experimentos concernientes a la evaluación, especialmente por las ventajas que presenta[10] y por ser un tipo de experimentación común en el estudio de índices P2P.
- Encontramos que principalmente se emplea el modelo de transitoriedad a nivel de nodo. Éste puede describirse mediante las distribuciones Weibull, Pareto y exponencial, siendo la primera la que mejor describe el fenómeno de la transitoriedad [Stu06]. Sin embargo, en [Zho09] se menciona que la distribución probabilística que se emplee no afecta mucho a los resultados.
    - Inherente al modelo de transitoriedad que se utilice, se hace uso de alguna medida para cuantificar la transitoriedad. Tanto el tiempo de vida medio, como la tasa de transitoriedad y el porcentaje de abandonos presentan algunos inconvenientes y no encontramos alguna otra medida que sea escalable y mida directamente la transitoriedad del sistema, tomando en cuenta tanto llegadas como salidas.
- Solo en [He07] realizan un análisis de confiabilidad tomando en cuenta la heterogeneidad de los pares, mencionada en [Stu06].
- En [Li04] se realiza un análisis paramétrico, pero no se mide la confiabilidad.
- El punto de quiebra, definido en [Zhi06], no es empleado en otra investigación siendo que esta medida puede usarse para comparar índices P2P.

Respaldándonos en los puntos anteriores y para los fines de nuestra investigación, utilizamos la *razón de búsquedas exitosas* y el *punto de quiebra* para medir la confiabilidad.

Con respecto al modelo de transitoriedad, consideramos que el *modelo de transitoriedad a nivel de nodo* es una buena opción debido a sus características de escalabilidad y a que es el más empleado en la literatura revisada.

Para los experimentos descritos en la siguiente sección usamos Oversim [oversim] como herramienta de simulación y modelamos la transitoriedad usando la variante de transitoriedad a nivel de nodo[11].

Dados los inconvenientes que poseen las medidas de transitoriedad empleadas, proponemos la tasa de transitoriedad normalizada, que escala en los experimentos y, según la definición de transitoriedad presentada en la sección 2.4, considera las entradas y salidas del sistema midiéndolas directamente.

[10] Ver 4.1.2.

[11] Para mayor detalle sobre Oversim consulte el apéndice A.1

Capítulo 5

Evaluación de la confiabilidad de índices P2P

Existen distintos índices P2P, cada uno con diferencias en su espacio de claves, estado, protocolo de encaminamiento y protocolo de autoorganización. El diseño de las dos primeras características determina la flexibilidad de la estructura distribuida. La tercera determina la eficiencia de las búsquedas. El protocolo de autoorganización es el mecanismo para mantener coherentes los apuntadores de la estructura. De las características antes mencionadas depende el correcto funcionamiento de un índice que es implementado en un sistema que sufre transitoriedad.

Debido a que el nivel de variabilidad de la membresía es diferente para cada sistema, consideramos que es importante evaluar la confiabilidad de los índices P2P para conocer qué tan bien realizan sus tareas y cuál es la cantidad de transitoriedad que toleran para hacerlas. Retomando de la introducción, presentamos los siguientes objetivos y metodología.

5.1. Objetivos

Objetivo general:

- Analizar la confiabilidad de los índices P2P en presencia de alta transitoriedad.

Objetivos particulares:

1. Comprender algunas de las propuestas de índices P2P y los mecanismos que utilizan para mantener la coherencia ante la transitoriedad.
2. Conocer las medidas existentes para cuantificar la confiabilidad y transitoriedad.
3. Definir un marco de evaluación de confiabilidad para índices P2P.
4. Evaluar la confiabilidad de los índices P2P.

5.2. Metodología:

1. Investigación de las características generales de los sistemas P2P.
2. Exploración del funcionamiento de índices P2P.

3. Identificación, definición y validación de medidas sobre evaluación de índices P2P (estado del arte).

4. Selección de la plataforma de simulación.

5. Especificación de un marco de evaluación para índices P2P.

6. Selección y evaluación de índices P2P.

A continuación, explicamos los pasos cuatro y cinco en las secciones 5.3 y 5.4 respectivamente. *En el anexo A se encuentra la información concerniente al paso cuatro. En el mismo anexo incluimos los detalles del simulador y del modelo de simulación. En la secciones venideras se expresa de manera resumida el marco de simulación.*

5.3. Tasa de transitoriedad normalizada

En 4.1.5 presentamos las medidas que emplea la comunidad para medir algún aspecto del dinamismo del sistema. Cabe resaltar que ninguna cuantifica todos los aspectos, de acuerdo a la definición manejada en esta investigación en la sección 2.4, del fenómeno de la transitoriedad.

- El tiempo de sesión medio, que comúnmente se usa en el modelo de transitoriedad a nivel de nodo, mide el mismo nivel de transitoriedad independientemente del cardinal de la membresía del sistema. Sin embargo, esta medida no cuantifica de manera directa la cantidad de pares que llegan o salen del sistema.

- La tasa de transitoriedad global λ, propia del modelo de transitoriedad por proceso de Poisson, mide directamente las entradas y salidas de los pares pero no escala en los experimentos. Por ejemplo, considerando una tasa promedio $\lambda = 10\ pares/minuto$ implica el $10\,\%$ de la membresía de un sistema con 100 pares, mientras que solo implica el $0.1\,\%$ de la membresía de un sistema con $10,000$.

- El porcentaje de abandonos, del modelo de transitoriedad por porcentaje de abandonos, es una medida que no depende del tamaño de la red y cuantifica directamente las salidas del sistema. Sin embargo, la medida solo considera los abandonos y deja de lado las llegadas.

Para mejorar los inconvenientes de las medidas anteriores, proponemos utilizar la medida *tasa de transitoriedad normalizada TTN*, que definimos a continuación.

Def. **Tasa de transitoriedad normalizada.** *Porcentaje de cambio en la membresía de un sistema P2P por unidad de tiempo ocasionado por las entradas y salidas de los pares. Se expresa en* $[\,\%/tiempo]$.

En seguida mostramos el análisis de la TTN. En los apartados 5.3.1, 5.3.2 y 5.3.3 se explica el cómo obtener la TTN, un análisis sobre la elección del tamaño de ventana de medición y la validación de la escalabilidad de la medida.

5.3.1. Obtención de la TTN

Para medir la *tasa de transitoriedad normalizada* de un experimento es necesario conocer el porcentaje de cambio que ocurre en el sistema durante una ventana de tiempo i de duración τ, donde τ es constante. Para ello se debe efectuar la siguiente operación:

$$(\,\%\text{ de cambio})_i = \frac{E_i + S_i}{N_i} * 100, \tag{5.1}$$

donde E es el número de entradas, S el número de salidas, N el número de pares que hubo en una ventana de tiempo i.

Después de realizar I mediciones continuas se debe obtener el porcentaje de cambio promedio.

$$(\,\%\text{ de cambio promedio}) = \frac{\sum_{i=1}^{I}(\,\%\text{ de cambio})_i}{I} \tag{5.2}$$

La *TTN* es el cociente del promedio del porcentaje de cambio sobre τ como se expresa en la siguiente fórmula:

$$TTN = \frac{\sum_{i=1}^{I}(\,\%\text{ de cambio})}{I} * \frac{1}{\tau} \tag{5.3}$$

5.3.2. Análisis del tamaño de la ventana de tiempo

Para observar el comportamiento del tamaño de la ventana de medición de la TTN capturamos muestras del porcentaje de cambio cada 1, 10 y 100 ms, un segundo, y 5, 10, 20 y 30 min bajo el siguiente escenario de simulación. Los detalles de esta simulaciones y las posteriores se encuentran en A.1.1.

- Características de los pares:
  - Tiempo de sesión descrito con la distribución Weibull con parámetro de forma $k = 0.5$; valor medio: 360 segundos.
- Fases de simulación:
  - Inicio: creación de 100 pares a una tasa de 10 pares por segundo.
  - Transición: 10 minutos.
  - Mediciones: 2 horas.

Tabla 5.1: Análisis de tamaño de ventana, nivel de confianza del 95 %.

Tam. de ventana τ	**% de cambio promedio**
$0.001\ s$	$575.85 \cdot 10^{-6} \pm 8.36 \cdot 10^{-6}$
$0.01\ s$	$5.76 \cdot 10^{-3} \pm 0.084 \cdot 10^{-3}$
$0.1\ s$	$57.58 \cdot 10^{-3} \pm 0.83 \cdot 10^{-3}$
$1\ s$	$575.9 \cdot 10^{-3} \pm 8.37 \cdot 10^{-3}$
$5\ min$	173.07 ± 2.52
$10\ min$	345.95 ± 5.09
$20\ min$	683.16 ± 10.03
$30\ min$	1021.04 ± 21.72

La tabla 5.1 muestra el porcentaje de cambio promedio con distintos tamaños de ventana τ. Se puede observar que el tamaño de la ventana no afecta a la TTN, ya que se obtiene el mismo nivel de transitoriedad independientemente del valor de τ. Por ejemplo, al convertir la TTN de la última fila $(1021.04\,\%)/(30\ min)$, se obtiene una tasa de $\sim 0.5\,\%/s$, que es el valor aproximado de la TTN de la columna cuatro; lo anterior se observa también con las demás mediciones. En el resto de las simulaciones usaremos una ventana de tiempo τ de $1\ s$.

5.3.3. Análisis de escalabilidad

En este apartado se muestra que la TTN mide la misma cantidad de transitoriedad independientemente del número de pares que integren la membresía del sistema. Cabe recordar que los experimentos que realizamos son escalables ya que usamos el modelo de transitoriedad a nivel de nodo con N variable.

Empleamos el siguiente escenario de simulación para el análisis de escalabilidad la TTN:

- Características de los pares:
  - Tiempo de sesión descrito con la distribución Weibull con parámetro de forma $k = 0.5$; valor medio: $60, 300, 600, 1200, 1800, 2400, 3000, 3600\ s$.
- Fases de simulación:
  - Inicio: creación de 100 y $1,000$ pares a una tasa promedio de 10 pares por segundo.
  - Transición: 10 minutos.
  - Mediciones: 2 horas.

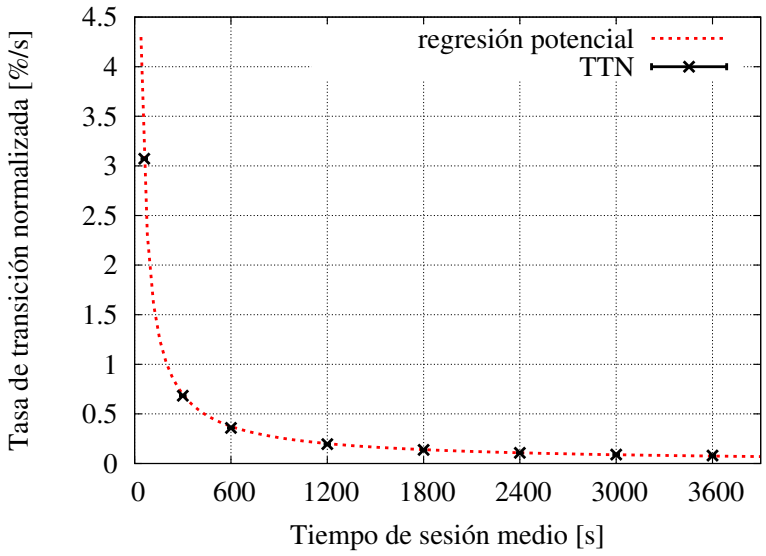
En la figura 5.1 se ilustra el comportamiento de la TTN contra el tiempo de sesión medio de los pares con un nivel de confianza del $95\,\%$. La incertidumbre es tan pequeña que no se aprecia en las gráficas.

Aplicamos regresión potencial para comparar la similitud de los valores que obtuvimos. Los coeficientes de las funciones tienen una diferencia aproximada de 1.5, que es un valor relativamente pequeño con respecto a la magnitud del coeficiente de las funciones. Con respecto al exponente, observamos que éste varía a partir del tercer decimal. En ambas curvas se presentan dos asíntotas; la TTN tiende a cero conforme aumenta el tiempo de sesión de los pares, mientras que cuando el tiempo de sesión de los pares tiende a cero, la TTN se dispara a infinito. Estas dos asíntotas son de poco interés para nuestra investigación. Cuando los tiempos de sesión medio son mayores a $2026\ s$ corresponden a la asíntota que tiende a cero, la cual describe un sistema poco dinámico; la que tiende a infinito presenta un dinamismo que no encontramos en la literatura y que implica que el tiempo de sesión medio de los pares es menor a 90 segundos.

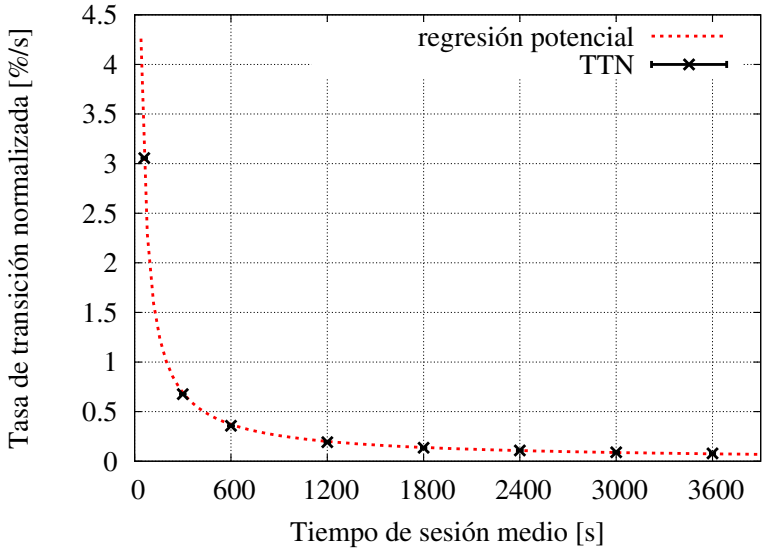
Con base en la observación anterior, identificamos un intervalo de tiempos de sesión que provoca mayor dinamismo en el sistema, a éste le llamamos *intervalo de alta transitoriedad*:

Def. **Alta transitoriedad.** *Es el intervalo de tiempos de sesión que provoca una tasa de transitoriedad normalizada de* 0.125 *a* $2\,\%/s$.

Si se emplea la función $114.29 \cdot x^{-0.895}$, el intervalo de alta transitoriedad ocurre cuando los tiempos de sesión de los pares están entre $91.6464s$ y $2026.8026s$. Para ejemplificar el efecto en el sistema de la alta transitoriedad, supongamos el valor más pequeño del intervalo, $TTN = 0.125\,\%/s$. Bajo tal dinamismo, la membresía cambia tantos pares como su cardinal promedio en aproximadamente 800



(a) membresía inicial de 100 pares, $115.78 \cdot x^{-0.896}$



(b) membresía inicial de 1000 pares, $114.29 \cdot x^{-0.895}$

Figura 5.1: TTN vs. tiempo de sesión medio, regresión potencial. Intervalo de confianza 95 %.

s, ya sea por entradas o salidas. De manera numérica[1], un sistema bajo una $TTN = 0.125\,\%/s$ con membresía promedio de $1,000$ pares sufre $1,000$ cambios en $\sim 800\ s$. Para el caso de $TTN = 2\,\%/s$, en $\sim 50\ s$ un sistema experimenta tantos cambios en su membresía como su cardinal promedio. Cabe aclarar que los cambios descritos en los ejemplos no implican una renovación total de la membresía, para ello se necesitaría que el total de pares, en cierto instante, salieran del sistema y que entraran $1,000$ pares nuevos; nuestra medida no es consciente de la identidad de los pares.

Por el uso de nuestra medida, redefinimos el *punto de quiebra* descrito en [Zhi06] para adaptarlo a nuestra investigación:

[1] A lo largo de este documento usaremos el símbolo "$\sim$" para indicar valores aproximados.

***Def.* Punto de quiebra.** *Si se modifica en promedio x porcentaje de pares por unidad de tiempo de la membresía de un sistema P2P y esta modificación ocasiona la falla del 50 % de las búsquedas generadas aleatoriamente, entonces x es la TTN que define el punto de quiebra del sistema.*

Con base en los resultados descritos en esta sección, comprobamos que la TTN es una medida escalable ya que en nuestros experimentos encontramos valores similares de ésta al aumentar de 100 a $1,000$ el cardinal inicial de la membresía. Adicionalmente, nuestra medida describe directamente la transitoriedad del sistema ya que mide el impacto de las entradas y salidas de los pares sobre la membresía promedio del sistema.

5.4. Análisis de confiabilidad

En esta sección se presenta el estudio correspondiente a la confiabilidad de la implementación del simulador de los índices P2P Chord, Pastry y Kademlia mediante el siguiente marco de evaluación constituido por tres fases:

1. *Evaluación estática por defecto.* El objetivo de ésta es medir la confiabilidad de los índices P2P usando parámetros de configuración similares a los recomendados por los autores de los índices.

2. *Evaluación estática paramétrica.* En esta fase se proponen un conjunto de valores para el estado y protocolo de autoorganización de los índices. El objetivo es encontrar qué configuración, entre las analizadas, garantiza mayor confiabilidad y un consumo razonable de ancho de banda.

3. *Evaluación con población heterogénea.* Se mide la confiabilidad considerando clases de pares cuyo tiempo de sesión medio es corto, medio y largo [Stu06, He07]. Los índices se ajustan con la configuración identificada en la fase anterior.

En las tres fases se aplica una prueba de encaminamiento basado en claves, donde cada par realiza una búsqueda de manera aleatoria por una clave o un par dado de alta en el sistema P2P[2]. A continuación presentamos la primera fase de evaluación.

5.4.1. Evaluación estática por defecto

Como un primer acercamiento analizamos la confiabilidad de los índices P2P configurándolos con los parámetros que vienen por defecto en la herramienta de simulación Oversim. Esto se debe a que las configuraciones del simulador son similares, y en algunos casos iguales, a las que proponen los autores de los índices P2P que seleccionamos.

El escenario de simulación que usamos en esta fase tiene la siguiente configuración:

1. Características de los pares:

   a) Tiempo entre búsquedas distribuido normalmente $(\mu = 60, \sigma = 6)[s]$. Solo se consideran valores positivos.

2. Características del sistema:

   a) $TTN = (0.125, 0.25, 0.5, 0.75, 1, 1.25, 1.5, 1.75, 2)[\,\%/s]$.

   b) Tiempo de sesión medio descrito con distribución Weibull con parámetro de forma $k = 0.5$.

[2]En el apéndice A se detalla la prueba.

c) Cardinal inicial de la membresía: 100 y $1,000$ pares.

3. Tipo de encaminamiento: iterativo[3]

4. Fases de simulación:

   a) Inicio: creación de pares a una tasa promedio de 10 pares por segundo.
   b) Transición: 10 minutos.
   c) Mediciones: 2 horas.

La configuración de los índices que tiene Oversim por defecto es la siguiente:

- Chord:
  - Lista de sucesores de tamaño ocho, valor próximo al propuesto en [Sto01] de $O(\log(N))$,
  - tiempo de ejecución del protocolo de estabilización: $20\ s$ y
  - tiempo de ejecución del protocolo de reparación: $30\ s$.
- Pastry:
  - Claves en base hexadecimal y
  - cardinal del conjunto de hojas $|H| = 16$, ambos valores iguales a los propuestos en [Ro01].
- Kademlia[4]:
  - Número de búsquedas paralelas $\alpha = 3$,
  - tamaño de los $buckets$ $k = 8$ y
  - actualización de $buckets$ cada $1000s$.

En la figura 5.2 se presentan las curvas de confiabilidad obtenidas al someter a Chord, Kademlia y Pastry a distintas tasas de transitoriedad y número de pares. A continuación damos nuestra interpretación de los resultados.

Kademlia

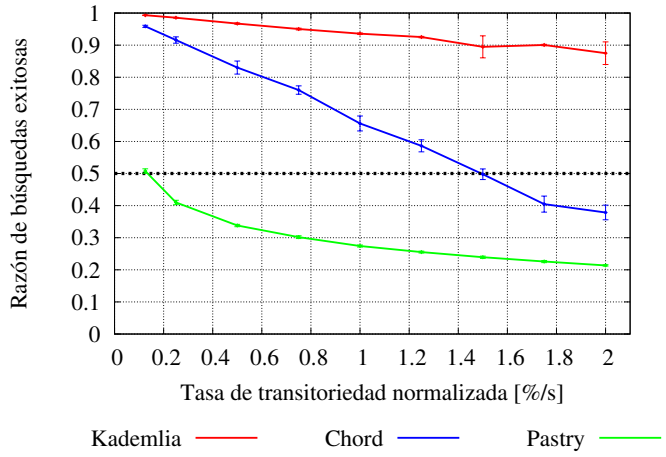
En 5.2(a) y 5.2(b) se observa que las curvas que describen la confiabilidad de Kademlia disminuyen lentamente cuando la TTN aumenta y que ambas no tienen punto de quiebra.

Para el escenario con membresía inicial de 100 pares, gráfica 5.2(a), la confiabilidad del índice se mantiene por encima del $85\,\%$. Específicamente, cuando la transitoriedad es de $0.125\,\%/s$ medimos una confiabilidad cercana al $100\,\%$. A partir de $TTN = 1.5\,\%/s$ la confiabilidad está por debajo $90\,\%$ y para la TTN más alta desciende a $\sim 86\,\%$.

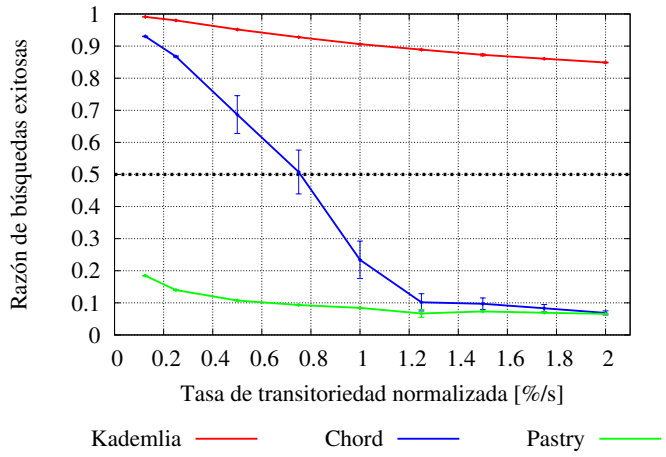
Con respecto al escenario con membresía inicial de $1,000$ pares, en la gráfica 5.2(b) se aprecia que la confiabilidad de Kademlia es mayor al $80\,\%$. Con $TTN = 0.125\,\%/s$ medimos una confiabilidad próxima al $100\,\%$ y que desciende al $90\,\%$ cuando $TTN = 1\,\%/s$. En el caso extremo, registramos una confiabilidad de $\sim 83\,\%$.

[3]El tipo de encaminamiento es la forma con la que se intercambian los mensajes. El encaminamiento iterativo consiste en que el par que genera una consulta interviene en todo el proceso de comunicación. En el encaminamiento recursivo la consulta es reenviada por los pares intermedios al objetivo y el par generador solo se entera del resultado de ésta. Existen híbridos entre estos tipos [Rt].

[4]Solo el número de búsquedas paralelas es acorde al valor propuesto en [Ma02]. Con respecto al tamaño de los *buckets*, en [Ma02] se sugiere 20 y que la actualización de éstos se realice cada hora.



(a) membresía inicial de 100 pares



(b) membresía inicial de 1000 pares

Figura 5.2: Análisis de confiabilidad de Chord, Kademlia y Pastry para 100 y $1,000$ pares. Nivel de confianza del 95 %.

Chord

La confiabilidad de Chord decrece conforme aumenta la TTN. Podemos observar en 5.2(a) que la confiabilidad de los experimentos con membresía inicial de 100 pares está en el rango de $(37, 96)\,\%$. Los extremos del intervalo anterior ocurren para $TTN = 2\,\%/s$ y $TTN = 0.125\,\%/s$ respectivamente. Para este escenario, el punto de quiebra de Chord es $TTN \approx 1.5\,\%/s$.

En los experimentos con membresía inicial de $1,000$ pares observamos que la confiabilidad de Chord desciende con mayor rapidez. Con $TTN \in [0.125, 1.25]\,\%/s$ la confiabilidad presenta un rango de $(10, 93)\,\%$. Con transitoriedad mayor a $1.25\,\%/s$ la confiabilidad se mantiene por debajo del $10\,\%$. En este escenario el punto de quiebra de Chord es $TTN \approx 0.75\,\%/s$.

Pastry

Para Pastry observamos poca confiabilidad ante la alta transitoriedad. En los resultados con membresía inicial de 100 pares, la confiabilidad desciende lentamente dentro del rango $(22, 51)\,\%$; en consecuencia el punto de quiebra $TTN \approx 0.125\,\%/s$.

En el escenario con $1,000$ pares, la confiabilidad de Pastry desciende lentamente en el rango $(6, 19)\,\%$, por lo que el índice no soporta la alta transitoriedad.

De manera general, observamos que la confiabilidad de los tres índices desciende cuando la TTN aumenta y cuando la membresía inicial crece en un orden. El caso más claro ocurre con el índice Chord, cuyo punto de quiebra desciende de $\sim 1.5\,\%/s$ a $\sim 0.75\,\%/s$. Lo anterior se debe a que el tiempo de estabilización de los índices aumenta cuando el tamaño del sistema aumenta.

Al contrastar los tres índices podemos observar que la confiabilidad de Chord desciende dentro de un rango con longitud mayor a la de los otros dos índices, lo que implica que el índice es poco estable ante la alta transitoriedad. La confiabilidad de Pastry y Kademlia decrece en un rango con longitud menor, por lo que estos índices son más estables.

Esta fase de evaluación de confiabilidad proporciona conocimiento sobre el comportamiento de los índices ante alta transitoriedad. Sin embargo, no tiene en cuenta el consumo de ancho de banda ni otras configuraciones de los índices que pueden dar mayor confiabilidad ante el escenario de simulación al que son expuestos.

5.4.2. Evaluación estática paramétrica

Los índices P2P poseen diferentes parámetros ajustables de los que depende su correcto funcionamiento. En consideración a las observaciones de la fase anterior, para esta evaluación ajustamos de manera estática diferentes parámetros del estado y protocolo de autoorganización de Chord y Kademlia. Buscamos alguna configuración que haga que el índice permanezca con una confiabilidad cercana al $100\,\%$ a sabiendas del ancho de banda que requiere el índice con dicha configuración. En esta parte de la evaluación y en la siguiente solo consideramos los índices que toleran la alta transitoriedad, es decir, Chord y Kademlia.

Las simulaciones que realizamos en esta fase tienen un escenario similar al de la fase uno con excepción de dos cambios:

- Fase de transición: 20 minutos.
- Cardinal inicial de la membresía: $1,000$ pares.

El fundamento del primer cambio es darle a los índices mayor tiempo de estabilización. La segunda diferencia se debe a nuestro interés por mejorar la confiabilidad de los sistemas con membresía inicial de $1,000$ pares, ya que es menor que en los sistemas con 100 pares.

Los resultados que mostramos en esta fase corresponden a una ejecución por cada configuración propuesta. Los promedios que manejamos corresponden a algún conjunto de configuraciones con algún parámetro común.

Kademlia

En esta fase, configuramos el índice Kademlia variando sus parámetros como se indica a continuación:

- Número de búsquedas paralelas $\alpha = 1, 3, 5, 10$,

- tamaño de los $buckets$ $k = 8, 16, 32$ y
- tiempo de ejecución del protocolo de estabilización de $buckets$ $stab = 600, 1800, 3600$ s.

En las figuras 5.3, 5.4 y 5.5 se ilustra la confiabilidad y consumo de ancho de banda de las distintas combinaciones de α, k y $stab$ que proponemos. En las gráficas, únicamente mostramos la confiabilidad y el ancho de banda en mps/s para los valores extremos y el valor intermedio del intervalo que definimos como *alta transitoriedad*, es decir, $TTN = 0.125, 1, 2\,\%/s$. Cada punto en las gráficas corresponde a una configuración, donde una configuración es una combinación de α, k y $stab$; hay 36 posibles.

Con respecto a la cantidad de mensajes que cada par envía, en la figura 5.3 se aprecia que mayor α implica mayor consumo de ancho de banda. Específicamente, en 5.3(a), 5.3(b) se observa en los recuadros que se forman agrupaciones en torno a α. Las agrupaciones se hacen menos visibles conforme disminuye la cantidad de transitoriedad; ésto ocurre a partir de $TTN = 0.5\,\%/s$. Con la menor tasa de transitoriedad que establecimos, en la figura 5.3(c) solo se distingue el grupo de configuraciones que tienen como parámetro $\alpha = 1$.

En cuanto a confiabilidad, en la figura 5.3 se observa que invertir ancho de banda en el número de búsquedas paralelas aumenta la confiabilidad, pero dependiendo del nivel de transitoriedad la inversión puede ser excesiva. Por ejemplo, en la gráfica 5.3(a) que representa la TTN más alta, para las configuraciones con $\alpha = \{1, 3, 5, 10\}$, la confiabilidad promedio es de $\sim 38, 79, 86, 92\,\%$ respectivamente, por lo que es necesario configurar al menos $\alpha = 5$ para que el índice tenga más del $80\,\%$ de confiabilidad.

Para el caso intermedio, es decir $TTN = 1\,\%/s$, en la gráfica 5.3(b) se observa que la confiabilidad aumenta ya que la mayoría de las configuraciones toleran la alta transitoriedad. Por tal motivo, se puede configurar α con un número menor a 5.

Al disminuir el nivel de transitoriedad también aumenta la confiabilidad, y para $TTN = 0.125\,\%/s$ tenemos que el total de configuraciones da más de $90\,\%$ de confiabilidad, como se aprecia en la gráfica 5.3(c).

El parámetro k aumenta la confiabilidad en las configuraciones con el mismo valor de α, pero no afecta en gran medida el consumo de ancho de banda. En la gráfica 5.4(a) se aprecia que las configuraciones con $k = 8$, encerradas en rectángulos, dan mayor confiabilidad en comparación con las que tienen $k = 16, 32$.

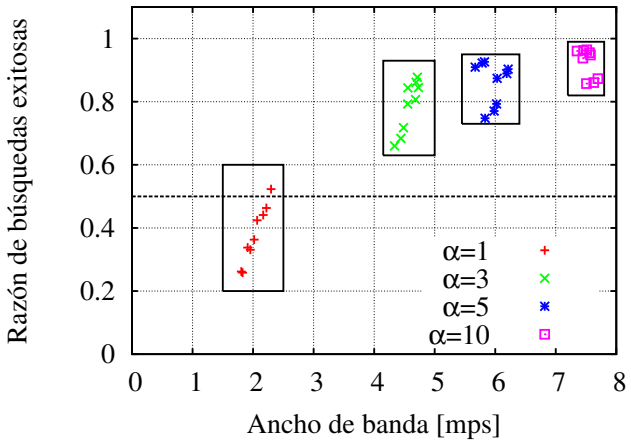
La tendencia anterior deja de percibirse mientras el nivel de transitoriedad desciende. Para $TTN = 1\,\%/s$, gráfica 5.4(b), las configuraciones con $k = 8, 16$ dan mayor confiabilidad que las que tienen $k = 32$. Numéricamente, en las configuraciones con $\alpha = 3$ encerradas en un rectángulo, tienen como confiabilidad promedio $\sim 91, 89, 82\,\%$ para $k = 8, 16, 32$ respectivamente.

Al igual que con el parámetro α, el tamaño k de los *buckets* deja de tener efecto sobre la confiabilidad mientras el nivel de transitoriedad disminuye, ya que como se observa en la gráfica 5.4(c), las configuraciones propuestas dan más del $90\,\%$ de confiabilidad.

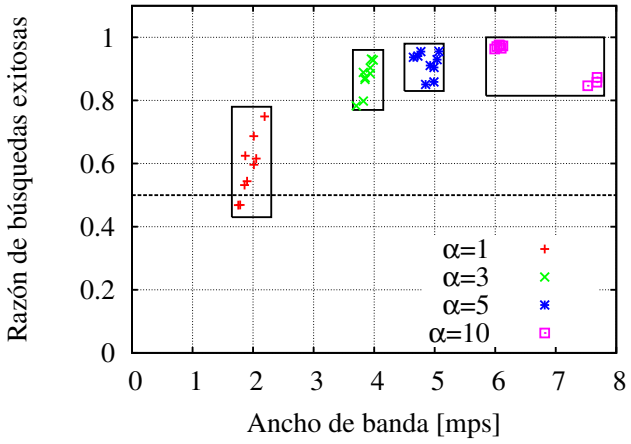
Manteniendo constantes los parámetros α y k, el parámetro $stab$ aumenta la confiabilidad si se configura con tiempos menores. Este parámetro tiene menor influencia en el consumo de ancho de banda. Cuando $TTN = 2\,\%/s$, gráfica 5.5(a), las configuraciones con $\alpha = 1$ presentan la menor confiabilidad, pero unicamente con $k = 8$ y $stab = 600$ s el índice no se quiebra.

En la gráfica 5.5(b) se sigue observando que $k = 600$ s mejora la confiabilidad, particularmente en las configuraciones encerradas en un rectángulo, que tienen en común $\alpha = 1$.

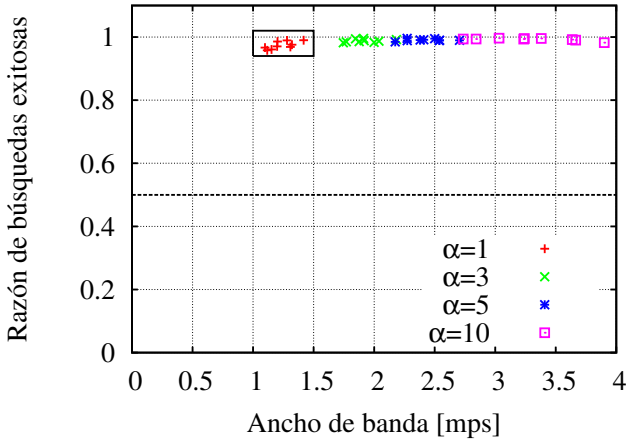
El aumento en el consumo de ancho de banda por el parámetro $stab$ se hace más notorio cuando disminuye el nivel de transitoriedad. Como se observa en la gráfica 5.5(c), las configuraciones $stab = 600$ s implican mayor envío de mensajes, para α y k fijas. De forma más notoria, lo anterior se percibe



(a) Transitoriedad de $2\,\%/s$

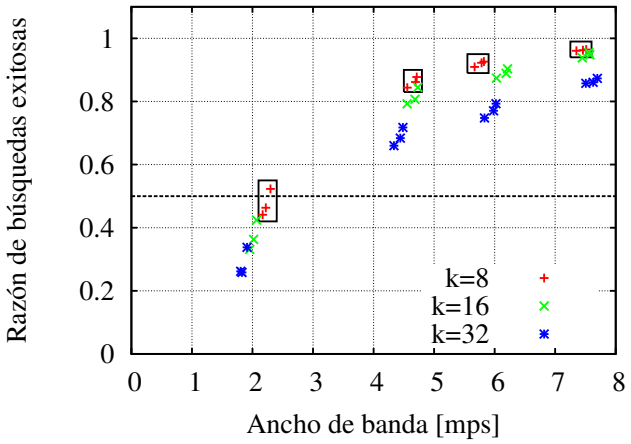


(b) Transitoriedad de $1\,\%/s$

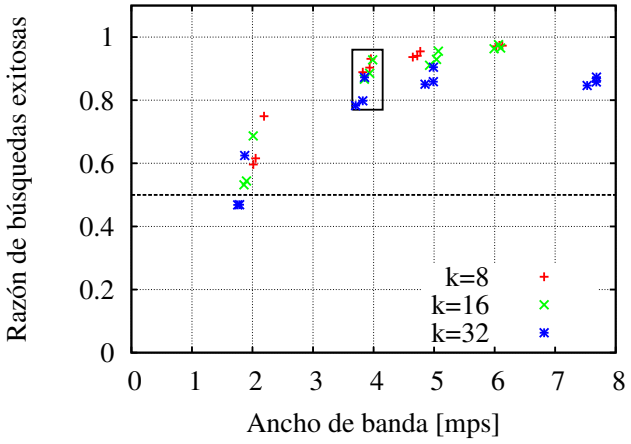


(c) Transitoriedad de $0.125\,\%/s$

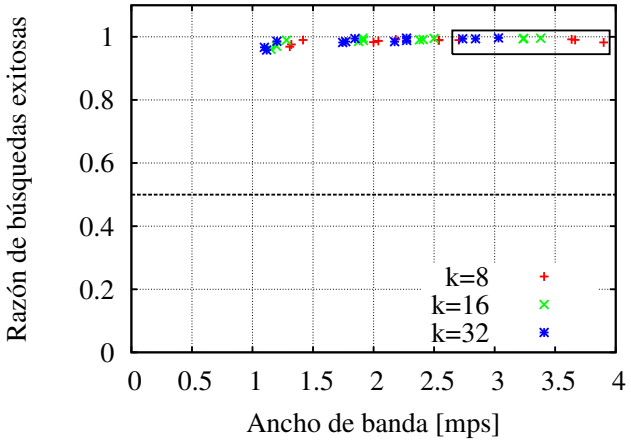
Figura 5.3: Kademlia. Análisis de confiabilidad y consumo de ancho de banda del parámetro α.



(a) Transitoriedad de $2\,\%/s$

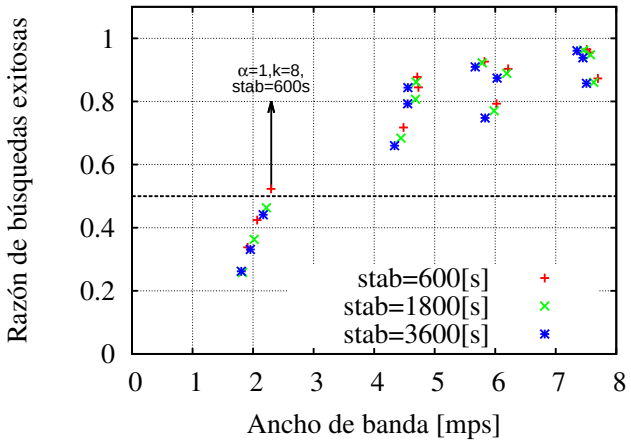


(b) Transitoriedad de $1\,\%/s$

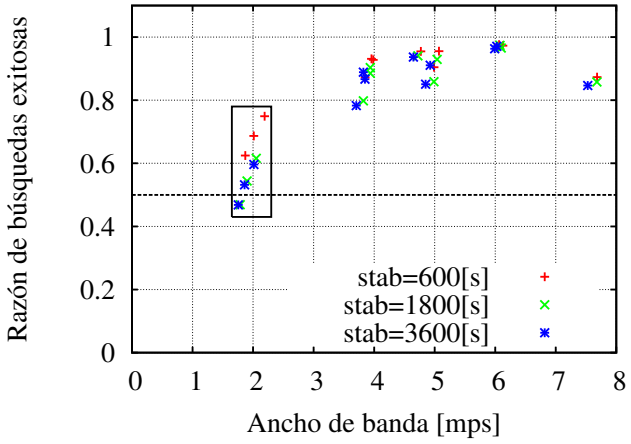


(c) Transitoriedad de $0.125\,\%/s$

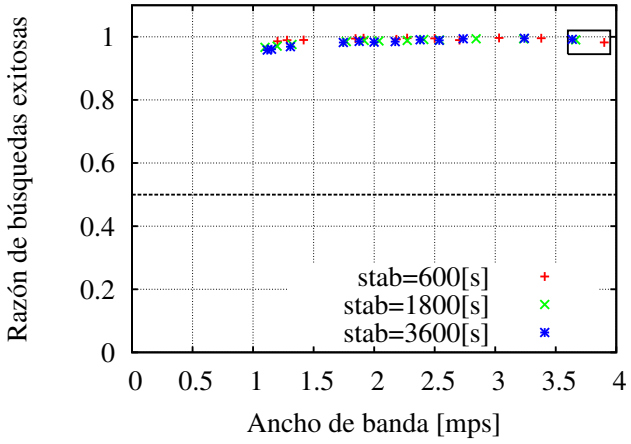
Figura 5.4: Kademlia. Análisis de confiabilidad y consumo de ancho de banda del parámetro k.



(a) Transitoriedad de $2\,\%/s$



(b) Transitoriedad de $1\,\%/s$



(c) Transitoriedad de $0.125\,\%/s$

Figura 5.5: Kademlia. Análisis de confiabilidad y consumo de ancho de banda del parámetro $stab$.

en las configuraciones encerradas en el rectángulo, que corresponden a $\alpha = 10$ y $k = 8$.

Del análisis anterior, observamos que Kademlia soporta la alta transitoriedad ya que solo ocho configuraciones tienen punto de quiebra igual a $TTN = 2\,\%/s$. Para $TTN \geq 0.75\,\%/s$ ninguna configuración quiebra al índice. Para la tasa de transitoriedad más baja, $TTN = 0.125\,\%/s$, la confiabilidad más baja es de $95.8\,\%$. Con relación a los parámetros, notamos que α determina en mayor parte la confiabilidad y el consumo de ancho de banda, seguido del parámetro k y $stab$.

La tabla 5.2 presenta las configuraciones que dan confiabilidad cercana al $100\,\%$ con base en el parámetro α, que es el que tiene mayor peso en cuanto a consumo de ancho de banda. Los parámetros k y $stab$ se fijan en 8 s y 600 s debido a que la mayoría de las configuraciones que proponemos con esos valores dan mayor confiabilidad en comparación con las otras de su tipo. En las columnas se observa que conforme disminuye la TTN la confiabilidad aumenta para cada α. Si analizamos las filas podemos observar que la confiabilidad aumenta conforme aumenta α, con excepción de $TTN \leq 0.25\,\%/s$.

Tabla 5.2: Kademlia. Confiabilidad, porcentaje de búsquedas exitosas para configuración $\alpha_i = \{1, 3, 5, 10\}$, $k = 8$ y $stab = 600s$

TTN $[\,\%/s]$	**Confiabilidad** $[\,\%]$			
	α_1	α_2	α_3	α_4
2	52.2	87.7	92.6	96.4
1.75	58.1	88.7	93.1	96.7
1.5	62.5	90.2	93.8	97
1.25	68.1	91.9	94.7	97.1
1.0	74.9	93.1	95.5	97.2
0.75	83.9	94.9	96.5	97.4
0.5	90.3	96.9	97.6	97.9
0.25	96.6	98.7	98.6	97.9
0.125	99.0	99.1	99.0	98.2

Con respecto al consumo de ancho de banda (AB), la tabla 5.3 describe la cantidad de mensajes y bytes que cada par envía por segundo. Las columnas muestran cómo se aminora el número de mensajes y bytes enviados conforme disminuye la TTN. En las filas se aprecia el incremento en el consumo de recursos al emplear un número de consultas paralelas mayor.

Chord

El índice Chord posee tres parámetros ajustables que afectan su comportamiento: el tamaño de la lista de sucesores y el tiempo de ejecución del protocolo de estabilización y el tiempo de ejecución del protocolo de reparación. No obstante, de [Sto01] se infiere que las búsquedas exitosas dependen de los parámetros relacionados con la lista de sucesores; para que la tabla de apuntadores permanezca sin errores, la lista de sucesores debe ser correcta. La tabla de apuntadores y su protocolo de reparación únicamente hacen eficiente al protocolo de encaminamiento.

Por lo anterior, en esta fase de evaluación de confiabilidad solo consideramos el ajuste del tiempo de ejecución del protocolo de estabilización y el tamaño de la lista de sucesores. Analizamos el índice

Tabla 5.3: Kademlia. Consumo de ancho de banda (AB) para configuración $\alpha_i = [1, 3, 5, 10]$, $k = 8$ y $stab = 600s$

TTN $[\,\%/s]$	**AB** $[mps]$				**AB** $[Bps]$			
	α_1	α_2	α_3	α_4	α_1	α_2	α_3	α_4
2	2.3	4.7	5.8	7.5	154.8	321	405	546.2
1.75	2.3	4.6	5.6	7.2	152.1	307.6	383.7	513.8
1.5	2.3	4.4	5.4	6.9	149.1	292.8	361.5	481.5
1.25	2.2	4.2	5.1	6.5	143.4	275.4	338.2	450.6
1.0	2.2	4.0	4.8	6.1	139.6	255.7	312.7	419.1
0.75	2.1	3.7	4.4	5.7	131.4	233.5	285.7	380.6
0.5	2.0	3.3	3.9	5.1	121.7	206.5	252.4	340.9
0.25	1.7	2.6	3.2	4.3	105.0	168.5	210.1	293.8
0.125	1.4	2.2	2.7	3.9	90.3	144.2	185.2	270.9

bajo la siguiente configuración:

- Lista de sucesores tamaño: 4, 8, 16 y 32 pares.
- Tiempo de ejecución del protocolo de estabilización: 5, 30, 60 y $300s$.

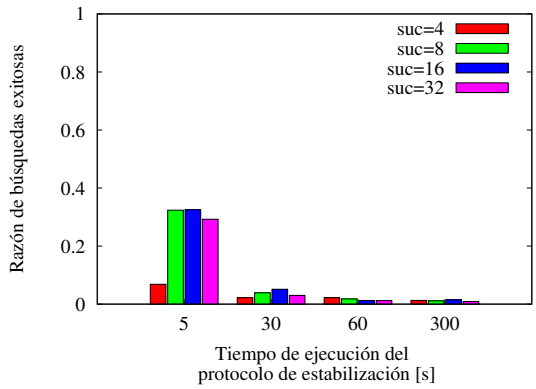
En la figura 5.6 se observa que la confiabilidad aumenta si se ejecuta a menor tiempo el protocolo de estabilización. En la gráfica 5.6(a) se observa que cualquier configuración quiebra al índice. Específicamente, con $stab = 5, 30, 60, 300\ s$ tienen una confiabilidad promedio de $\sim 25, 4, 2, 1\,\%$ respectivamente.

Al disminuir la TTN a $1\,\%/s$, gráfica 5.6(c), solo las configuraciones con $stab = 5\ s$ y $suc > 4$ están por encima del $50\,\%$ de confiabilidad. Las configuraciones con $stab = 30, 60, 300\ s$ quiebran al índice y tienen una confiabilidad promedio de $\sim 13, 4, 1\,\%$.

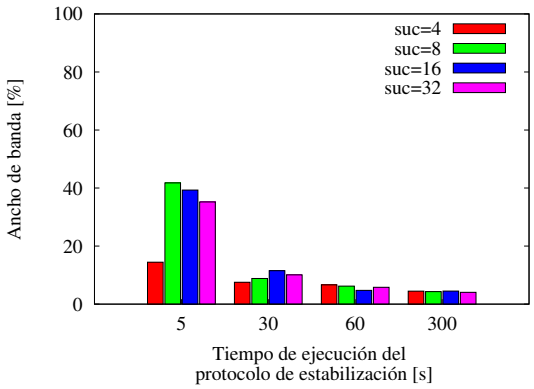
Para el menor valor del intervalo de alta transitoriedad, se ilustra en 5.6(e) que la mayoría de las configuraciones mantienen una confiabilidad superior al $60\,\%$ y que con $stab = 300\ s$ se quiebra el índice. Puntualmente, la confiabilidad promedio con $stab = 30, 60, 300\ s$ corresponde a $\sim 88, 59, 6\,\%$.

La influencia del tamaño de la lista de sucesores es difícil de caracterizar debido a que no se observa ningún comportamiento sistemático, con excepción en las configuraciones con $stab = 5\ s$. Por ejemplo, en la gráfica 5.6(e) las configuraciones que dan la confiabilidad más cercana al $100\,\%$ tienen $suc = 4$ y $stab = 5, 30\ s$, sin embargo, el patrón no se repite en las configuraciones con $stab = 60, 300\ s$. En las gráficas 5.6(e), 5.6(c) y 5.6(a), se puede apreciar en las barras correspondientes a las configuraciones con $stab = 5\ s$, que $suc = 8$ aumenta la confiabilidad, seguido por $suc = 16$ y $suc = 32$; con $suc = 4$ la confiabilidad desciende.

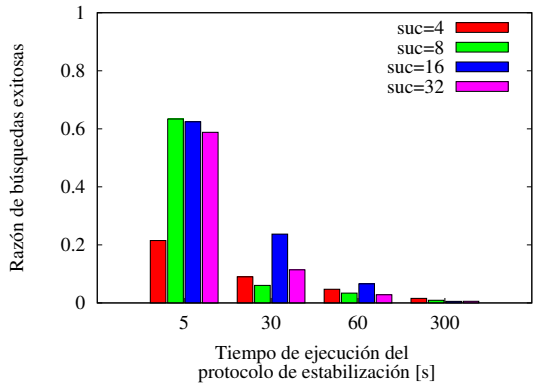
Con respecto al consumo de ancho de banda, en las gráficas 5.6(f), 5.6(d) y 5.6(b) se observa que el ancho de banda tiene un comportamiento semejante a la confiabilidad. Para hacer un mejor contraste, normalizamos el ancho de banda considerando que $\sim 5.6\ mps$ es el $100\,\%$. Por ejemplo, en las gráficas 5.6(e) y 5.6(f), con excepción de la configuración con $stab = 300\ s$ y $suc = 8$, se observa en el tamaño de las barras que la cantidad de confiabilidad determina el consumo de ancho de banda. Lo anterior también se aprecia en las gráficas 5.6(e) y 5.6(f), correspondientes a $TTN = 1, 2\,\%/s$.



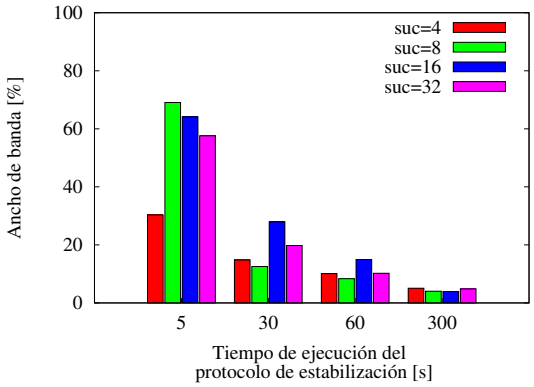
(a) $TTN = 2\,\%/s$. Confiabilidad.



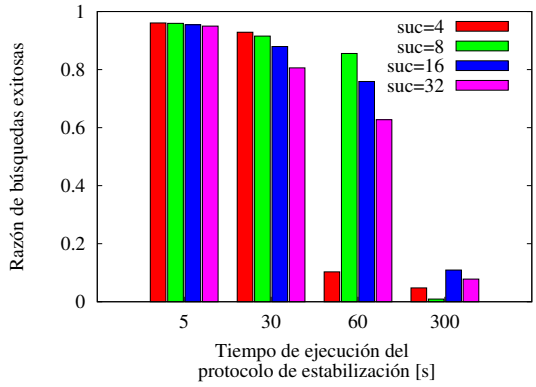
(b) $TTN = 2\,\%/s$. Ancho de banda.



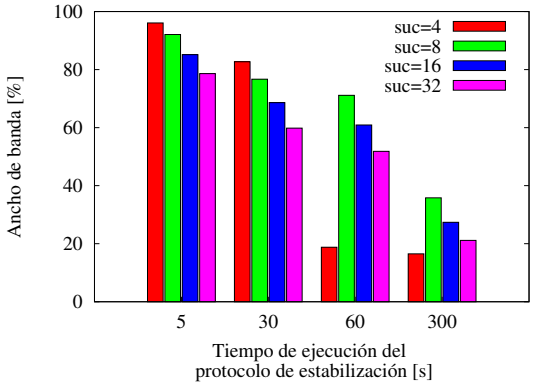
(c) $TTN = 1\,\%/s$. Confiabilidad.



(d) $TTN = 1\,\%/s$. Ancho de banda.



(e) $TTN = 0.125\,\%/s$. Confiabilidad.



(f) $TTN = 0.125\,\%/s$. Ancho de banda.

Figura 5.6: Chord. Análisis de confiabilidad y consumo de ancho de banda por tiempo de estabilización.

La relación entre el consumo de ancho de banda y la confiabilidad se debe a que, como se menciona en [Zhi06], la razón de búsquedas exitosas describe de manera indirecta la coherencia del índice. De otra manera, una alta confiabilidad implica que los enlaces entre los pares son correctos, por lo que se puede encaminar un mayor número de mensajes.

Otro punto a considerar es que la confiabilidad de Chord está subordinada a otro factor. En [Sto01],

precisamente en el teorema siete, se menciona que Chord depende de su estabilidad inicial para operar correctamente. Sin embargo, en el modelo de transitoriedad que empleamos los pares se crean con un tiempo de sesión definido. Lo anterior implica que existe transitoriedad con valores altos de TTN, sin importar que la simulación esté en la fase de inicio o transición[5].

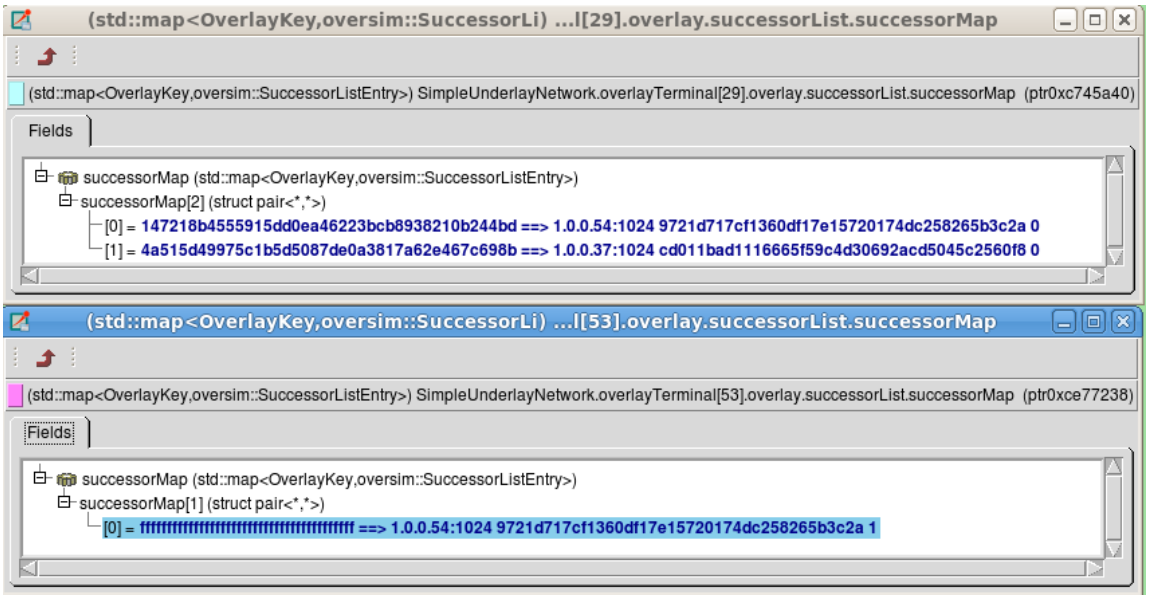


Figura 5.7: Lista de sucesores fallida del par con identificador 29.

La alta transitoriedad provoca que el anillo Chord se divida. La figura 5.7 muestra la lista de sucesores de los pares con identificador 29 y 53[6], de una simulación de Chord ejecutada de manera gráfica, con membresía inicial de 30 pares, lista de sucesores tamaño cinco y tiempo de ejecución del protocolo de estabilización de 30 segundos a una $TTN = 2\,\%/s$. La lista del par 29 está compuesta por los pares con dirección IP 1.0.0.37 y 1.0.0.54 respectivamente. Sin embargo, la lista del par con IP 1.0.0.54 es errónea y el par con IP 1.0.0.37 no está en el sistema, lo que implica que el anillo Chord está roto porque el par 29 no conoce a sus sucesores. Por las razones expuestas, la mayoría de las configuraciones que proponemos dan baja confiabilidad.

Tabla 5.4: Chord. Confiabilidad y consumo de ancho de banda para $stab = 5s$ y $suc = 8$.

Atributo	**Tasa de transición normalizada** $[\%/s]$								
	2	1.75	1.5	1.25	1.0	0.75	0.5	0.25	0.125
Confiabilidad $[\%]$	23.6	27.3	33.4	47.3	60.4	71.5	83.2	91.9	96.0
Ancho de banda $[mps]$	1.8	2.0	2.3	3.0	3.7	4.2	4.7	5.0	5.1
Ancho de banda $[Bps]$	156.7	175.5	217.2	228.5	254.0	304.2	329.6	320.0	329.2

La tabla 5.4 ilustra específicamente los resultados de confiabilidad de la configuración $stab = 5\ s$

[5] Ver apéndice A.1.
[6] Ver la cabecera de las ventanas.

y $suc = 8$ ante la alta transitoriedad. En la fila concerniente a la confiabilidad se aprecia cómo la confiabilidad mejora si se reduce la TTN. La tabla complementa el análisis de ancho de banda correspondiente a las gráficas 5.6(f), 5.6(d) y 5.6(b). Hay que considerar que tanto la cadencia de actualización de sucesores ($5\ s$), como la de reparación de apuntadores ($30\ s$), es mayor al tiempo medio de generación de búsquedas aleatorias, que es de $60\ s$, por lo que los mensajes de mantenimiento aportan una parte considerable del ancho de banda.

5.4.2.1. Comparación entre Chord y Kademlia

Los experimentos de esta fase muestran que Kademlia es un índice confiable para la mayoría de las configuraciones que proponemos pues solo ocho de ellas tienen punto de quiebra $TTN = 2\,\%/s$. Por el contrario, Chord muestra pocas configuraciones cuya confiabilidad esté por arriba del $50\,\%$. En particular, las configuraciones con $stab = 5\ s$ son más confiables.

El análisis de Kademlia también arroja que hay configuraciones que superan el $90\,\%$ de confiabilidad, independientemente de la TTN a la que se someta. En orden, los parámetros que muestran mayor influencia en la confiabilidad son el número de búsquedas paralelas, particularmente cuando $\alpha > 1$, el tamaño k de los $buckets$ y tiempo de ejecución del protocolo de estabilización. Otra variable que influye en la confiabilidad de este índice es la cantidad de tráfico [Ma02], ya que cada mensaje lleva adjunta información de mantenimiento.

Por otra parte, Chord requiere que la lista de sucesores no tenga errores, por lo que se debe ejecutar ansiosamente el protocolo de estabilización para mantener coherente el índice. El análisis paramétrico de este índice indica que para soportar la alta transitoriedad se debe ejecutar el protocolo de estabilización cada 5 segundos. El tamaño de la lista de sucesores no tiene un comportamiento que indique que contribuye a aumentar la confiabilidad.

En cuanto al consumo de ancho de banda, Kademlia envía más mensajes por par por segundo que Chord mientras $TTN \in [2, 1]\,\%/s$. Al disminuir la TTN, Kademlia consume menos ancho de banda que Chord. En promedio, la configuraciones de Kademlia con $\alpha = 10$ para $TTN = 0.125\,\%/s$ consumen $\sim 3.3\ mps$; las configuraciones de Chord con $stab = 5$ emplean $\sim 5.1\ mps$. Este comportamiento está influenciado por los mensajes de mantenimiento; en Kademlia los k-$buckets$ se actualizan cada $600, 1800, 3600\ s$, mientras que la lista de sucesores de Chord lo hace cada $5, 30, 60, 300\ s$.

La figura 5.8 muestra la confiabilidad de Chord y Kademlia bajo el intervalo de alta transitoriedad. Para Kademlia presentamos la configuración con $\alpha = 1$, $k = 8$ y $stab = 600\ s$; para Chord $stab = 5\ s$ y $suc = 8$. Justificamos esta comparación con base en que Chord no tiene implementado un mecanismo de búsquedas paralelas.

Gráficamente se observa que ambos índices presentan un comportamiento similar, pero la pendiente de Chord es más pronunciada que la de Kademlia. Para $TTN = 2\,\%/s$ la confiabilidad tiene una diferencia de $\sim 25\,\%$. Mientras la TTN disminuye, la diferencia en la confiabilidad se hace menor siendo que, cuando $TTN = 0.125\,\%/s$, aproximadamente $2\,\%$ separa a las curvas. También tenemos que el punto de quiebra de Chord ocurre cuando $TTN \approx 1.25\,\%/s$ y que Kademlia apenas sobrepasa el $50\,\%$ de confiabilidad cuando $TTN = 2\,\%/s$.

La ventaja que presenta Chord, con respecto a la primera fase de evaluación, es que supera su punto de quiebra, ocurriendo a una $TTN \approx 1.25\,\%/s$ con la configuración presentada; el punto de quiebra para el análisis anterior ocurre cuando $TTN \approx 0.75\,\%/s$.

Al abstraer el número de búsquedas paralelas, las diferencias en los protocolos de autoorganización de Chord y Kademlia están en los mecanismos de actualización de estado por lo que, con base en los resultados expuestos, los mecanismos de Kademlia hacen que el índice resista la alta transitoriedad. Además, debido al comportamiento similar de las dos curvas, se infiere que el número de consultas paralelas contribuye a mejorar la coherencia en un índice P2P ya que cuando $\alpha > 1$ la confiabilidad

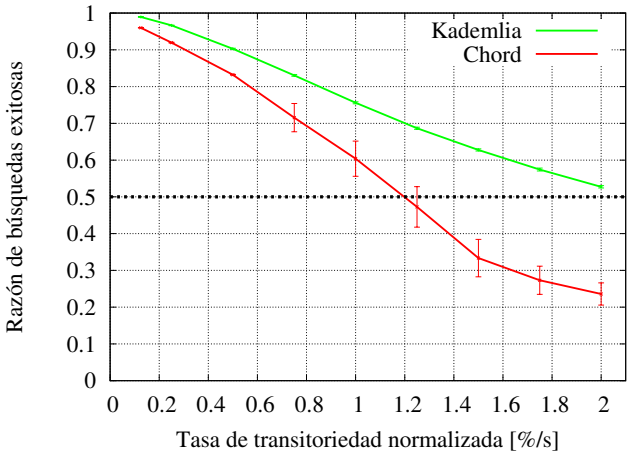


Figura 5.8: Contraste de confiabilidad entre Chord y Kademlia. Nivel de confianza del 95 %.

aumenta.

5.4.3. Evaluación con población heterogénea

Los estudios de confiabilidad en los que se simulan poblaciones de pares con tiempos de sesión homogéneos son únicamente un caso particular. Stutzbach et ál. [Stu06] y Herrera et ál. [He07] mencionan que es importante considerar que hay pares que exhiben un comportamiento similar y que pueden se agrupados respecto al tiempo de sesión. Por ejemplo, en los estudios *in situ* de Stutzbach et ál., se menciona que un porcentaje considerable de pares permanece por largo tiempo y una menor parte tiene un tiempo sesión corto. Por otro lado, Herrera et ál. define en su marco de evaluación de sistemas P2P una clasificación de los pares según su tiempo de sesión que citamos a continuación.

- *Benefactores*: Son aquellos pares que permanecen conectados largos períodos, del orden de días.
- *Comunes*: Pares cuyo tiempo de sesión está determinado por el tiempo de descarga necesario para obtener algún recurso.
- *Mirones*: Su tiempo de sesión es muy pequeño, del orden de minutos.

El objetivo de esta fase de evaluación es medir la confiabilidad de los índices en una población de pares heterogénea y observar si éstos se benefician con la presencia de pares benefactores. Para ello proponemos someter a los índices a la transitoriedad generada por las tres clases de pares mencionadas en dos diferentes escenarios, con cardinal inicial de la membresía de $1,000$ pares.

En la tabla 5.5 se presenta las características de la población heterogénea que examinamos. De manera condensada expresa lo siguiente:

- Los benefactores no se retiran del sistema y su proporción corresponde al $10\,\%$, $20\,\%$ y $30\,\%$ de la población.
- Los comunes integran la mayor parte de la población con una proporción del $60\,\%$.
- El número de mirones se determina según en el número de benefactores, siendo que cuando los benefactores son Be, y los comunes Co, los mirones son $100\,\% - Be - Co$. Por ejemplo, con $10\,\%$ de benefactores tenemos $30\,\%$ de mirones.

Tabla 5.5: Escenarios de simulación para población heterogénea.

Experimento	**Composición** $[\%]$			**Tiempo de sesión** $[s]$		
	$\%Be$	$\%Co$	$\%Mi$	$\%Be$	$\%Co$	$\%Mi$
Esc. 1	30	60	10	∞	1800	300
	20	60	20	∞	1800	300
	10	60	30	∞	1800	300
Esc. 2	30	60	10	∞	~ 934	~ 92
	20	60	20	∞	~ 934	~ 92
	10	60	30	∞	~ 934	~ 92

Usamos la TTN para conocer la cantidad de transitoriedad que generan las distintas clases de pares en los experimentos. El valor que mostramos es el promedio obtenido de la ejecución de veinte simulaciones.

A partir de los resultados de la fase dos, configuramos a Chord y Kademlia de la siguiente manera:

- Chord
  - Lista de sucesores $suc = 8$ y
  - cadencia de estabilización $stab = 5\ s$.
- Kademlia
  - Tamaño de $buckets$, $k = 8$,
  - cadencia de estabilización de $stab = 600\ s$ y
  - número de búsquedas paralelas, $\alpha = 5$.

Realizamos las simulaciones con las características del escenario de la fase dos pero definimos que la duración de la fase de mediciones fuera de 4 horas.

5.4.3.1. Población heterogénea: primer escenario

Este escenario simula un sistema P2P donde el tiempo de descarga de los recursos es de aproximadamente $1800\ s$. Por esa razón, el tiempo de sesión promedio de los comunes es de 30 minutos.

Tabla 5.6: Resultados de confiabilidad para escenario 1. Nivel de confianza del $95\,\%$.

Composición			**Confiabilidad [%]**		**TTN Global** $[\%/s]$
$\%Be$	$\%Co$	$\%Mi$	Kademlia	Chord	
30	60	10	98.68 ± 0.05	95.38 ± 0.15	~ 0.14
20	60	20	98.74 ± 0.04	93.15 ± 0.09	~ 0.21
10	60	30	99.04 ± 0.06	90.97 ± 0.15	~ 0.28

La gráfica 5.9 muestra la comparación de la confiabilidad obtenida de dar a los benefactores, comunes y mirones las proporciones y tiempos de sesión descritos en la columna correspondiente al

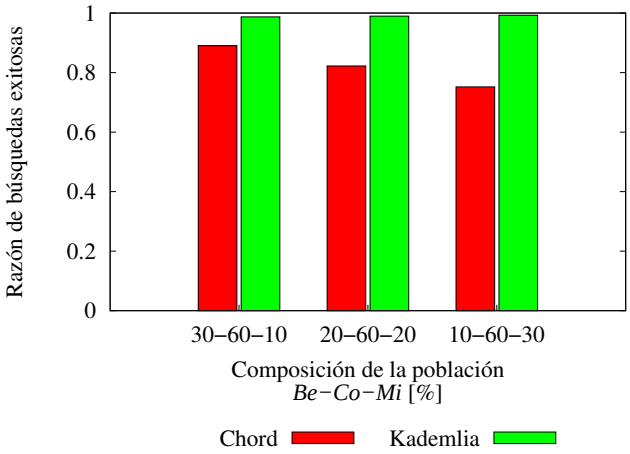


Figura 5.9: Análisis de confiabilidad sobre población heterogénea, escenario 1, poca transitoriedad global.

escenario uno de la tabla 5.5. La altura de las barras muestra que la confiabilidad de ambos índices sobrepasa el $90\,\%$, en consecuencia, los índices no se quiebran. También se puede apreciar que Chord presenta la menor razón de búsquedas exitosas en los tres experimentos.

En la tabla 5.6 se muestra que las simulaciones de este escenario generan varias TTN que aumentan con respecto al porcentaje de mirones. El valor de transitoriedad más alto es $TTN = 0.27\,\%/s$ y corresponde al tercer experimento. Tal TTN implica que la membresía cambia $1,000$ pares en ~ 370 s.

La confiabilidad de Kademlia, descrita en la tabla 5.6, es mayor al $98\,\%$ en los tres experimentos. Para Chord, mientras la TTN aumenta, la confiabilidad disminuye del $95.3\,\%$ al $90.9\,\%$. Estos resultados son similares a los presentados en las tablas 5.4 y 5.2, específicamente para los valores correspondientes a $TTN = [0.125, 0.25]\,\%/s$.

5.4.3.2. Población heterogénea: segundo escenario

Para aumentar la cantidad de transitoriedad, asignamos tiempos de sesión menores para los comunes y mirones, de $934.6252s$ y $91.6464s$ respectivamente. Este escenario simula un sistema donde los recursos pueden obtenerse rápidamente, como música en formato comprimido. Los tiempos de sesión que definimos para estos experimentos corresponden a $TTN = 0.25\,\%/s$ para la clase de los comunes y a $TTN = 2\,\%s$ para la de los mirones.

Tabla 5.7: Resultados de confiabilidad para escenario 2. Nivel de confianza del $95\,\%$.

Composición			**Confiabilidad [%]**		**TTN Global** [$\%/s$]
$\%Be$	$\%Co$	$\%Mi$	Kademlia	Chord	
30	60	10	98.71 ± 0.02	89.01 ± 0.11	~ 0.36
20	60	20	98.96 ± 0.04	82.23 ± 0.14	~ 0.58
10	60	30	99.29 ± 0.03	75.2 ± 0.18	~ 0.8

En los experimentos 4, 5 y 6 se generan distintas tasas de transitoriedad que están en función de la

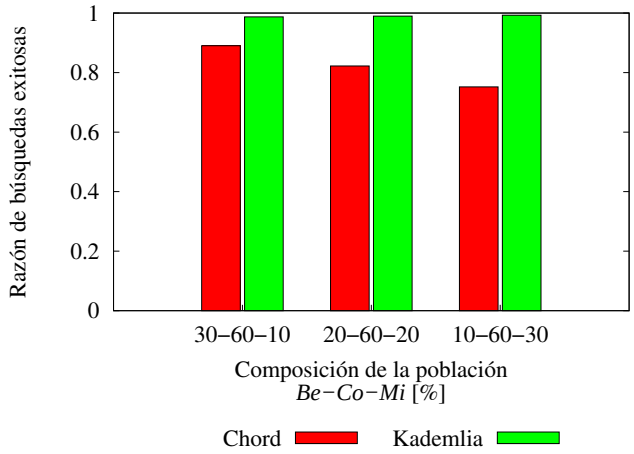


Figura 5.10: Análisis de confiabilidad sobre población, escenario 2, media transitoriedad.

composición del sistema. Al igual que en el escenario anterior, a mayor porcentaje de mirones mayor TTN. En la tabla 5.7, para el experimento 4 se registra una $TTN = 0.3597\,\%/s$ que implica que la membresía del sistema tiene $\sim 1,000$ cambios cada ~ 4.5 minutos. En los experimentos 5 y 6, el fenómeno ocurre cada ~ 3 y ~ 2 minutos respectivamente.

En la figura 5.10 presentamos de manera gráfica la confiabilidad que se obtiene al aumentar la transitoriedad en una red con población heterogénea. Podemos observar que Kademlia tiene una confiabilidad mayor o igual al $98\,\%$ en los casos examinados.

Al comparar la confiabilidad descrita en la tabla 5.7 con la de la tabla 5.2, se observa que una mejora en la confiabilidad, ya que ésta se mantiene en $\sim 98\,\%$; con población homogénea y misma transitoriedad desciende a $\sim 96\,\%$. En este caso se aprecia el efecto de los benefactores ya que, aun con solo $10\,\%$ de ellos, Kademlia se muestra estable a pesar del incremento de la transitoriedad.

El índice Chord presenta el mismo comportamiento decreciente que en el conjunto de experimentos con población homogénea, siendo que registramos una confiabilidad similar a la presentada en la tabla 5.4. El diseño de Chord considera que todos los pares tienen el mismo comportamiento, por lo que la lista de sucesores no tiene en cuenta el tiempo de duración de las sesiones. Por esta razón, la confiabilidad de Chord no se mejora con la presencia de benefactores.

5.5. Sumario

En este capítulo se presenta un estudio y comparación sobre confiabilidad de los índices P2P Chord, Kademlia y Pastry. En primer lugar mostramos una medida para cuantificar la transitoriedad en sistemas P2P. En seguida, evaluamos la confiabilidad de los índices a través de un marco de evaluación conformado por tres fases.

Con respecto a la medida, la cual llamamos *tasa de transitoriedad normalizada* TTN, encontramos en las simulaciones que ésta supera los problemas de otras medidas ya que cuantifica directamente la cantidad de entradas y salidas que realizan los pares y también escala en los experimentos. Además, encontramos que nuestra medida registra una cantidad similar de transitoriedad independientemente del tamaño de la red. También, del análisis de la TTN, definimos el intervalo *alta transitoriedad* que describe los tiempos de sesión con los que se genera un mayor cambio en la membresía del sistema.

Posteriormente realizamos el análisis concerniente a la confiabilidad de los índices P2P. En un primer estudio medimos la confiabilidad de los índices, configurándolos con los parámetros que vienen

por defecto en el simulador y obtuvimos que Kademlia presenta mayor confiabilidad, Chord se quiebra con los valores superiores al extremo inferior de la TTN y Pastry no soporta la alta transitoriedad. También observamos que la confiabilidad desciende cuando el tamaño de la red aumenta.

Para poder comparar a los índices de manera equitativa realizamos un estudio paramétrico, sometiendo a los índices a alta transitoriedad con diferentes configuraciones de algunos de sus parámetros ajustables. Encontramos que Chord y Kademlia poseen configuraciones que mejoran mucho su confiabilidad. Sin embargo, Chord no alcanza la confiabilidad de Kademlia.

Por último, analizamos la confiabilidad de los índices sometiéndolos a la transitoriedad generada por una población de pares heterogénea. Los experimentos muestran que la confiabilidad de Chord disminuye al incrementarse la TTN global y que, con Kademlia, la confiabilidad se mantiene constante. Por esta observación notamos que Kademlia mejora su confiabilidad al tener en cuenta a los benefactores, en cambio, la de Chord no mejora ya que no los considera en su diseño.

En la sección 3.4 presentamos un contraste sobre los mecanismos de los índices que influyen en la resistencia a la transitoriedad. Con base en esos puntos y en los resultados que obtuvimos observamos que:

- El mecanismo de búsquedas paralelas, en conjunto con el mecanismo de acarreo a cuestas, aumentan la confiabilidad. Estos mecanismos son un factor clave debido a que, en las mediciones concernientes a Kademlia encontramos que la mayoría de las configuraciones con $\alpha > 3$ resisten la alta transitoriedad.

- Con base en los experimentos con población heterogénea, es recomendable considerar el tiempo de sesión de los pares, ya que si los enlaces lógicos, determinados por el estado de los pares, se construyen teniendo en cuenta a los pares cuyo tiempo de sesión es considerablemente mayor se mantendrá la coherencia del índice.

- La topología de anillo es menos resistente ante la alta transitoriedad que la topología de árbol; como se observa cuando la lista de sucesores de algún par Chord es totalmente incorrecta, o cuando todo el conjunto de hojas falla.

- El enfoque perezoso de detección de fallas, como el de Pastry, no es efectivo ante la alta transitoriedad; el enfoque ansioso implica invertir más ancho de banda a pesar de que la transitoriedad disminuya, como en Chord.

Capítulo 6

Conclusiones y trabajo futuro

Este proyecto de investigación presentamos un análisis de la confiabilidad de los índices P2P en un ambiente de alta transitoriedad siguiendo objetivos que mostramos a continuación:

1. Comprender algunas de las propuestas de índices P2P y los mecanismos que utilizan para mantener la coherencia ante la transitoriedad.
2. Conocer las medidas existentes para cuantificar la confiabilidad y transitoriedad.
3. Definir un marco de evaluación de confiabilidad para índices P2P.
4. Evaluar la confiabilidad de los índices P2P.

En principio, elaboramos un compendio de conceptos básicos sobre sistemas P2P para entender sus características y la naturaleza con la que operan. Posteriormente, estudiamos el funcionamiento de tres índices P2P (Chord, Kademlia y Pastry), y examinamos sus características: espacio de claves, estado, protocolo de encaminamiento y protocolo de autoorganización.

Después, realizamos un estado del arte sobre los esfuerzos que ha hecho la comunidad para medir la confiabilidad de los índices P2P. En este punto, encontramos varias medidas que se pueden usar para obtener la confiabilidad de un índice ante transitoriedad. Para nuestra investigación adoptamos una variante del *punto de quiebra*[1] [Zhi06] y la razón de búsquedas exitosas. También identificamos varios modelos con diferentes niveles de abstracción y medidas para calibrar la transitoriedad a la que se somete a un índice.

Para realizar nuestra experimentación, y con base en el estado del arte, recurrimos a la simulación. Para ello indagamos sobre las características y ventajas de varias plataformas de simulación; nos inclinamos por utilizar Oversim [oversim].

En la etapa de desarrollo, encontramos que las medidas descritas en el estado del arte sobre transitoriedad no dan una descripción del fenómeno con respecto a la definición de transitoriedad que presentamos en esta investigación. Así que propusimos una medida a la que llamamos *tasa de transitoriedad normalizada* (TTN), ésta cuantifica el porcentaje de cambio en la membresía de un sistema P2P por unidad de tiempo. La TTN tiene la ventaja de ser independiente del número de pares que componen la red. El análisis de nuestra medida proporciona un intervalo en el que la transitoriedad representa mayor cambio en el número de pares en la red, a éste lo llamamos intervalo de *alta transitoriedad*, $TTN \in [0.125, 2]\,\%/s$. En seguida definimos un marco de evaluación de confiabilidad de índices consistente de tres fases: evaluación estática por defecto, evaluación estática paramétrica y evaluación con población heterogénea.

[1] La TTN necesaria para que el $50\,\%$ de las búsquedas fallen. Ver definición en sección 5.3.3.

Para el primer punto, realizamos varias simulaciones considerando 100 y $1,000$ pares en el sistema. En esta evaluación Kademlia mostró mayor confiabilidad seguido por Chord y Pastry. Con respecto al *punto de quiebra*, Kademlia no lo presentó, Chord se quebró ante una TTN intermedia en el intervalo de alta transitoriedad y Pastry no soportó la alta transitoriedad. Los índices analizados vieron disminuida su confiabilidad en los experimentos con cardinal inicial de $1,000$ pares.

El segundo conjunto de experimentos consistió en buscar qué configuración garantiza mayor confiabilidad entre un conjunto de valores propuesto por nosotros para un sistema con cardinal inicial de $1,000$ pares y considerando el ancho de banda. Realizamos una simulación por cada configuración para cada índice. En los resultados concernientes a Kademlia, encontramos que la confiabilidad depende en gran medida del número de consultas paralelas. Con respecto a Chord, encontramos que el parámetro de mayor peso en la confiabilidad es la cadencia de ejecución del protocolo de estabilización. Además, comparamos la configuración con mayor confiabilidad obtenida de Chord y Kademlia. Haciendo abstracción del número de consultas paralelas, encontramos que la confiabilidad de estos índices es similar, sin embargo, Chord no supera a Kademlia.

Por último, usando la configuración identificada en la fase anterior, sometimos a Chord y Kademlia a la transitoriedad generada por una población de pares con tiempos de sesión heterogéneos. Los resultados arrojaron que Kademlia mantiene su confiabilidad constante y alta con únicamente $10\,\%$ de pares con tiempo de sesión muy largo (benefactores). La confiabilidad de Chord decrece conforme la TTN global aumenta y no presenta mejora con la presencia de benefactores. En las simulaciones correspondientes a esta parte, ninguno de los dos índices presentó punto de quiebra.

En general y por los resultados obtenidos de nuestro marco de evaluación, Kademlia da mayor confiabilidad que los otros índices analizados; las características de su estado y protocolo de autoorganización lo hacen más resistente a la transitoriedad, en comparación a Chord y Pastry.

Trabajo futuro

Las perspectivas que abre este trabajo de investigación son las siguientes:

- Mejorar los mecanismos de los índices analizados. Por una lado, una posible vía para aumentar la resistencia de Chord y Pastry es utilizar mecanismos similares a los de Kademlia. Por otro lado, para los tres índices analizados, es posible ajustar de forma dinámica los parámetros del estado y protocolo de autoorganización.

- Usar otros modelos de transitoriedad y medir el impacto que el modelo tiene en los resultados de confiabilidad de índices P2P.

- Estudiar las características y confiabilidad de otros índices P2P, como CAN, Koorde, Kelips, Tapestry, Viceroy y Bamboo para así observar si poseen propiedades, diferentes a las de Chord, Pastry y Kademlia, que aporten mayor resistencia a la transitoriedad.

- Con las características encontradas, proponer un índice que mejore a los ya existentes.

- Con el fin de configurar la transitoriedad de manera comprensible, y con base en la medida *tasa de transitoriedad normalizada*, proponer un modelo de transitoriedad que considere el conocimiento existente con respecto al dinamismo de los sistemas P2P.

- Proponer un marco de evaluación de confiabilidad y consumo de ancho de banda para sistemas P2P no estructurados.

- Comparar la confiabilidad y consumo de ancho de banda entre sistemas P2P estructurados y no estructurados.

Apéndice A

Simuladores P2P

Este apéndice resume los criterios de selección de la herramienta de simulación que se empleó en esta investigación y también muestra su funcionamiento y algunos detalles técnicos de ésta.

En la actualidad, existe una amplia variedad de simuladores de sistemas P2P como P2Psim [P2Psim], NetHawk East [NetHawk], PlanetSim [Pu09], PeerSim [Mo09], Oversim [Ba09] y otros [Na07].

Elegimos comparar, bajo el criterio de [Na07], PlanetSim, PeerSim y Oversim porque eran las herramientas más novedosas. El criterio de [Na07] consiste en contrastar las arquitectura, escalabilidad, facilidad de uso y abstracción de red subyacente de los simuladores. Por los fines de nuestra investigación, agregamos los puntos estadísticas, protocolos incluidos y modelo de transitoriedad. A continuación presentamos la descripción de éstos.

Arquitectura Describe el diseño y funcionalidad del simulador y si se puede establecer el comportamiento del nodo. También se menciona si el simulador está basado en eventos discretos o ciclos.

Escalabilidad Especifica la cantidad de nodos que puede soportar el software de simulación

Facilidad de uso Detalla qué tan simple es utilizar la herramienta de simulación, si el simulador tiene un API común [Da03] que permita que el código sea fácilmente implementado, comprensible, alterado y portado a otros simuladores, si el código está bien documentado y qué tan simple es seguir la documentación, así como el lenguaje de programación.

Simulación de red subyacente Este criterio menciona si el simulador solo hace una abstracción de la red superpuesta o hace realista la topología subyacente, es decir, si puede simular la capa de red, dando características a enlaces y paquetes, tráfico de red y retardos.

Estadísticas Es de gran importancia que los resultados que arroja el simulador sean fáciles de manipular y de ser llevados a programas que permitan el análisis estadístico.

Protocolos incluidos Lista los protocolos P2P que vienen integrados con el simulador.

Modelo de transitoriedad Menciona si es posible implementar un modelo de transitoriedad en la red y que tipo de modelo puede ser implementado.

Propiedad	PlanetSim	PeerSim	OverSim
Arquitectura	simulador de eventos discretos, soporta redes estructuradas y no-estructuradas	simulador de eventos discretos y basado en rondas para redes estructuradas y no-estructuradas, se puede establecer el comportamiento de los pares	simulador de eventos discretos, soporta redes estructuradas y no-estructuradas y se puede configurar el comportamiento de los pares
Escalabilidad	10^5 nodos	10^6 nodos con simulación basada en ciclos	10^5 nodos
Facilidad	basado en la C-API, escrito en java, extensa documentación, los autores ofrecen soporte y el proyecto posee un repositorio de subversión	escrito en java, posee abstracción de grafos que pueden ser exportados a formatos populares, extensa documentación, posee ejemplos de configuración	basado en la C-API, emplea el motor de simulación de OMNeT++, escrito en C++, extensa documentación, posee la interfaz gráfica de OMNeT++
Red Subyacente	permite cargar modelos de latencia (Brite, Pajek y GML), también se puede el ancho de banda de los enlaces	para la simulación basada en eventos, la capa de transporte puede ser modelada a través de conjuntos basados en trazas (conjunto del Rey)	Posee tres maneras de simular la red subyacente: Simple, SingleHost e INET siendo el más real el último, ya que permite modelar la capa MAC

Tabla A.1: Comparación sobre simuladores de redes P2P, puntos de [Na07].

A.1. Oversim

En las tablas A.1 y A.2 se expresa el desarrollo de los puntos de comparación antes mencionados. De su análisis se eligió Oversim [Ba09, oversim] porque posee varias posibilidades de simular la transitoriedad de redes P2P, por las estadísticas que se pueden obtener a través de la herramienta y porque usa como motor de simulación a OMNeT++[omnet]. También, porque está basado en la C-API [Da03] que esta compuesta por tres capas: encaminamiento basado en claves, interfaz de aplicación y aplicación. A continuación se explican estos tres puntos.

El encaminamiento basado en claves (capa 0) es el servicio de localización que se provee en las redes superpuestas. En esta capa se manejan los datos necesarios para encaminar mensajes. También posee tres funciones para encaminar mensajes entre pares. La primera se emplea para enviar un mensaje de un par solicitante a un primer par. La segunda se emplea para encaminar un mensaje y la última para entregar el mensaje a su destino. También se manejan otras funciones estrictamente locales cuyo objetivo es que la aplicación pueda acceder a los datos guardados en el estado de los pares.

La interfaz de aplicación (capa 1) está formada por tres abstracciones de la capa anterior. Estas son la tabla de dispersión distribuida (DHT *distributed hash table*), localización y encaminamiento de objetos descentralizados (DOLR *Decentralized Object Location and Routing*) y multidifusión (CAST

Propiedad	PlanetSim	PeerSim	OverSim
Estadísticas	se ofrecen estadísticas simples como número de mensajes empleados y tiempo total de simulación, las estadísticas extras pueden obtenerse mediante programación orientada a aspecto AOP	se tiene una clase observador que mide parámetros de grafos y por la cual se pueden programar la estadísticas deseadas	Tiene programado varias estadísticas como entrega de paquetes fallidas y exitosas, tráfico de red por nodo, cuenta de salto de paquetes, la salida puede ser procesado por gnuplot
Protocolos incluidos	Chord, Symphony, SkipNet, Gnutella	Pastry, Chord, Kademlia, Skipnet, BitTorrent	Chord, Kademlia, Pastry, Bamboo, Koorde, Broose, GIA
Modelo de transitoriedad	no implementado	no se especifica	basado en tiempo de sesión, en inter-llegadas, por trazas o llegadas y salidas aleatorias

Tabla A.2: Comparación sobre simuladores de redes P2P, puntos propuestos por nosotros.

multicast, anycast). La DHT es una simple interfaz que provee un almacén para guardar y recuperar objetos de cualquier clase. La interfaz DOLR provee un servicio de directorio descentralizado. CAST es un servicio que provee grupos escalables de comunicación y coordinación.

La capa de aplicación (capa 2) es la abstracción más alta. Puede emplear la capa uno como interfaz o emplear directamente el servicio ofrecido por la capa cero.

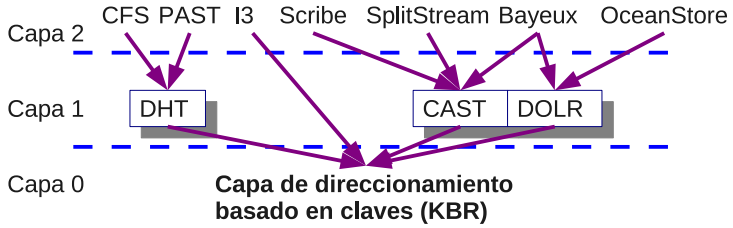


Figura A.1: Esquema de abstracción de API común [Da03].

La figura A.1 ilustra las abstracciones descritas anteriormente y además presenta algunas aplicaciones y el uso de las capas. CFS son las siglas de sistema de archivos de clúster (*cluster file system*). PAST [Dr01] es un aplicación distribuida de gran escala basada en Internet que sirve como almacén global; usa Pastry como interfaz. I3 [Sto04] es una aplicación que ofrece una abstracción basada en un *punto de cita* para superar las limitantes del modelo punto-a-punto de la Internet; emplea como base a Chord y, como se ve en la figura, usa directamente el servicio de la capa 0. SCRIBE [Cas02] y SplitStream [Cas03] son aplicaciones que ofrecen servicio multidifusión; están construidas sobre Pastry. Bayeux [Zhu01] también ofrece servicio multidifusión orientado a contenido multimedia. OceanStore [Rh01] es un almacén de datos de escala mundial. Los últimos dos usan como sustrato a Tapestry.

A continuación mostraremos algunas características y el funcionamiento de Oversim. Este simulador se define como un marco de simulación de redes par-a-par de código abierto que puede simular redes superpuestas estructuradas y no estructuradas. También posee una interfaz gráfica que puede ser

empleada para depuración de algoritmos.

Para hacer abstracción de la red subyacente[1] Oversim puede emplear tres modelos:

- *Simple*: Es el modelo por defecto del simulador. Los nodos se colocan en un espacio euclidiano n-dimensional donde la latencia se genera al calcular la distancia entre éstos. Las posiciones de los nodos se eligen para que sean similares a las medidas de latencia de Internet tomadas por el proyecto CAIDA/Skitter[2]. A los pares también se les agregan características como ancho de banda, retraso de acceso, pérdida de paquetes y una cola lógica.
- *INET*: Se emplea para simulaciones de redes de acceso heterogéneo, encaminadores del *backbone* y terminales móviles. Bajo esta abstracción la pila IP es completamente modelada.
- *SingleHost*: Esta abstracción actúa como *middleware* para ejecutar los protocolos de redes superpuestas sobre redes reales.

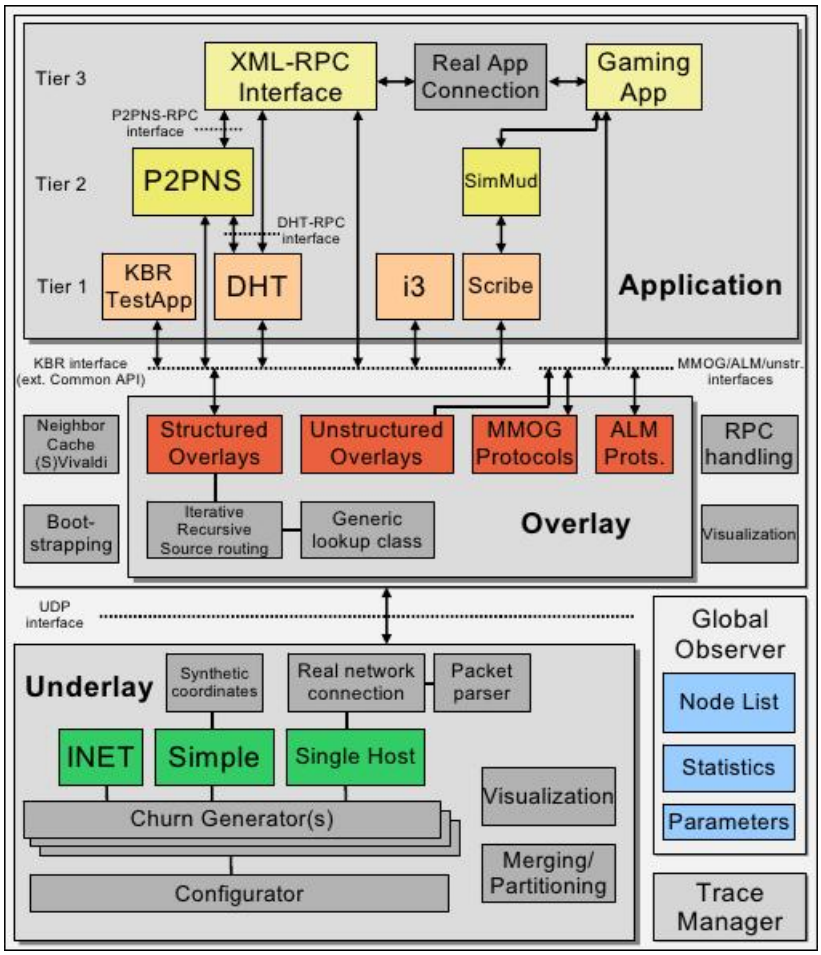


Figura A.2: Arquitectura modular de Oversim.

La figura A.2 ilustra la arquitectura de Oversim. En nuestro caso empleamos para las simulaciones el modelo de red subyacente *simple*. Para generar transitoriedad, empleamos el modelo basado en

[1] Siguiendo el modelo OSI, las características de la red subyacente estarían definidas de la capa de transporte a las capas inferiores.

[2] `http://www.caida.org/tools/measurement/skitter/`

tiempo de vida (*LifetimeChurn*)[3]. Además el simulador posee otros modelos llamados "sin transitoriedad" (*No Churn*), "transitoriedad Pareto" (*Pareto Churn*), "transitoriedad aleatoria" (*Random Churn*) y "transitoriedad basada en trazas (*Trace Churn*)".

Del módulo superior, correspondiente a la red superpuesta, nos centramos en las redes estructuradas. El simulador tiene los siguientes protocolos: Chord, Pastry, Kademlia, Bamboo y Koorde. Con respecto al módulo de encaminamiento, Oversim posee recursivo e iterativo, e híbridos entre éstos. Bajo nuestra experiencia, las simulaciones con encaminamiento recursivo son más lentas.

Respecto al módulo concerniente a la aplicación, nuestros experimentos se realizaron en la aplicación de pruebas del encaminamiento basado en claves (*KBR TestApp*). Bajo esta prueba, los pares generan consultas por claves aleatorias o nodos existentes. El receptor no regresa respuesta alguna, pero el emisor emplea un método que registra y analiza los datos de las consultas realizadas. Las consultas están distribuidas normalmente, tomando en cuenta solo valores positivos y tienen una media de $\mu = 60s$ y una desviación estándar de $\sigma = 60s/3$. La prueba no implica la recuperación de los recursos.

A.1.1. Modelo de simulación

Las simulaciones[4] se realizan en tres etapas:

- Fase de inicio: En ésta se crean X nodos cada t segundos en promedio. Al igual que en la prueba de encaminamiento basado en claves, el tiempo creación de los nodos está descrita por la distribución *trunc-normal* en donde solo se toman los valores positivos de la distribución[5]. Los nodos son creados con un tiempo de vida asignado, por lo que en esta fase ocurre transitoriedad. Para superar ese problema, el simulador crea más nodos que los N que le son asignados. La fase termina después de ejecutar N ciclos de creación.

- Fase de transición: El objetivo de la fase de transición es permitir que la red a examinar alcance cierta estabilidad antes de iniciar la fase de mediciones.

- Fase de mediciones: Durante la duración de ésta se capturan las estadísticas que son de importancia para el usuario.

Para compensar las salidas de los nodos, y debido a que éstos no regresan, después de que un nodo sale se deja pasar un tiempo muerto t_{dead}, descrito por la misma distribución de probabilidad usada en los tiempos de sesión.

Con respecto a la ejecución de las simulaciones, en Oversim, se leen dos archivos de configuración, uno se llamar `omnetpp.ini` y el otro `default.ini`. Tanto el primero como el segundo se escriben, con la misma sintaxis del simulador OMNeT++ [ManualOmnet]. En `omnetpp.ini` se escriben los parámetros claves de la simulación; en `default.ini` se escriben la mayoría de los parámetros que se deseen por defecto. Es importante aclarar que además de los parámetros de configuración que los autores manejan, en el simulador están implementados otros parámetros, los cuales se mantuvieron constantes para cualquier experimento y son los que vienen por defecto en el simulador.

[3] En la sección 4.1.4 le llamamos transitoriedad a nivel de nodo con N variable.

[4] La duración de tiempos que se establecen en estas fases corresponde a tiempo de simulación, no tiempo real, por lo que la duración real de los experimentos dependerá de la complejidad de los escenarios y del hardware donde se ejecute.

[5] `http://www.omnetpp.org/doc/omnetpp40/api//group__RandomNumbersCont.html`.

Referencias bibliográficas

[ManualOmnet] «Manual OMNeT++.», 2011. URL www.omnetpp.org/doc/omnetpp40/UserGuide.pdf.

[NetHawk] «Nethawk.», 2011. URL https://www.nethawk.fi/.

[omnet] «OMNeT++.», 2011. URL http://www.omnetpp.org/.

[oversim] «Oversim.», 2011. URL http://www.oversim.org/.

[P2Psim] «p2psim a simulator for a peer-to-peer protocols.», 2011. URL http://pdos.csail.mit.edu/p2psim/.

[Planet] «Planet lab.», 2011. URL http://www.planet-lab.org/.

[Rt] «Tipos de encaminamiento en Oversim.», 2011. URL http://www.oversim.org/wiki/OverSimKbrRouting.

[Ba09] BAUMGART, I., B. HEEP, y S. KRAUSE. «OverSim: A scalable and flexible overlay framework for simulation and real network applications.» En *Proceedings of the Ninth International Conference on Peer-to-Peer Computing*. 2009, P2P'09, 87–88. URL http://dx.doi.org/10.1109/P2P.2009.5284505.

[Bi07] BINZENHÖFER, A., y K. LEIBNITZ. «Estimating churn in structured P2P networks.» En *Proceedings of the 20th international teletraffic conference on Managing traffic performance in converged networks*. Springer-Verlag, Berlin, Heidelberg, 2007, ITC20'07, 630–641. URL http://dl.acm.org/citation.cfm?id=1769187.1769257.

[Cas04] CASTRO, M., M. COSTA, y A. ROWSTRON. «Performance and Dependability of Structured Peer-to-Peer Overlays.» En *Proceedings of the 2004 International Conference on Dependable Systems and Networks*. IEEE Computer Society, Washington, EUA, 2004, 9–18. URL http://dl.acm.org/citation.cfm?id=1009382.1009717.

[Cas02] CASTRO, M., P. DRUSCHEL, A. KERMARREC, y A. ROWSTRON. «Scribe: A large-scale and decentralized application-level multicast infrastructure.» *IEEE Journal on Selected Areas in Communication (JSAC)*, 20, nº 8, (2002), 1489–1499.

[Cas03] CASTRO, M., P. DRUSCHEL, A. M. KERMARREC, A. NANDI, A. ROWSTRON, y A. SINGH. «SplitStream: high-bandwidth multicast in cooperative environments.» *SIGOPS Operating Systems Review*, 37, (2003), 298–313. URL http://doi.acm.org/10.1145/1165389.945474.

[Da03] DABEK, F., B. ZHAO, P. DRUSCHEL, J. KUBIATOWICZ, y I. STOICA. «Towards a Common API for Structured Peer-to-Peer Overlays.» En *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems*. 2003, IPTPS03, 33,44.

[Dr01] DRUSCHEL, P., y A. ROWSTRON. «PAST: A Large-Scale, Persistent Peer-to-Peer Storage Utility.» *Workshop on Hot Topics in Operating Systems*, 0.

[SHA] EASTLAKE, D., y P. JONES. «US Secure Hash Algorithm 1 (SHA1).» *Inf. Téc. FIPS 180-1*, U.S. Department of Commerce/NIST, Springfield, EUA, April 1995.

[Kr02] GUMMADI, K., S. SAROIU, y S. GRIBBLE. «King: estimating latency between arbitrary internet end hosts.» En *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment*. ACM, Nueva York, EUA, 2002, IMW '02, 5–18. URL http://doi.acm.org/10.1145/637201.637203.

[Gu03] GUPTA, I., K. BIRMAN, P. LINGA, A. DEMERS, y R. RENESSE. «Kelips: Building an Efficient and Stable P2P DHT Through Increased Memory and Background Overhead.» En *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems*. 2003, IPTPS '03, 160–169. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.13.3464.

[Je09] GUSTEDT, J., E. JEANNOT, y M. QUINSON. «Experimental Validation in Large-Scale Systems: a Survey of Methodologies.» *Parallel Processing Letters*, 19, (2009), 399–418. URL http://hal.inria.fr/inria-00364180/en/.

[He07] HERRERA, O., y T. ZNATI. «Modeling Churn in P2P Networks.» En *Proceedings of the 40th Annual Simulation Symposium*. IEEE Computer Society, Washington, EUA, 2007, 33–40. URL http://dl.acm.org/citation.cfm?id=1249253.1250391.

[Ka03] KAASHOEK, F. M., y D. R. KARGER. «Koorde: A Simple Degree-Optimal Distributed Hash Table.» En *Proceedings of the 2nd International Workshop on Peer-to-Peer System*. 2003, IPTPS '03, 98–107. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.12.4430.

[Ka97] KARGER, D., E. LEHMAN, T. LEIGHTON, R. PANIGRAHY, LEVINE, y M. LEWIN. «Consistent hashing and random trees: distributed caching protocols for relieving hot spots on the World Wide Web.» En *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*. ACM, Nueva York, EUA, 1997, STOC '97, 654–663. URL http://doi.acm.org/10.1145/258533.258660.

[Li04] LI, J., J. STRIBLING, T. GIL, R. MORRIS, y F. KAASHOEK. «Comparing the performance of distributed hash tables under churn.» En *Proceedings of the 3rd International Workshop on Peer-to-Peer Systems*. IPTPS'04.

[Lua05] LUA, E. K., J. CROWCROFT, M. PIAS, R. SHARMA, y S. LIM. «A Survey and Comparison of Peer-to-Peer Overlay Network Schemes.» *IEEE Communications Surveys and Tutorials*, 7, (2005), 72–93. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.109.6124.

[Mal02] MALKHI, D., M. NAOR, y D. RATAJCZAK. «Viceroy: a scalable and dynamic emulation of the butterfly.» En *Proceedings of the twenty-first annual symposium on*

Principles of distributed computing. ACM, Nueva York, EUA, 2002, PODC '02, 183–192. URL http://doi.acm.org/10.1145/571825.571857.

[Ma02] MAYMOUNKOV, P., y D. MAZIÈRES. «Kademlia: A Peer-to-Peer Information System Based on the XOR Metric.» En *Revised Papers from the First International Workshop on Peer-to-Peer Systems*. Springer-Verlag, Londres, Reino Unido, 2002, IPTPS '02, 53–65. URL http://dl.acm.org/citation.cfm?id=646334.687801.

[Me08] MESHKOVA, E., J. RIIHIJÄRVI, M. PETROVA, y P. MÄHÖNEN. «A survey on resource discovery mechanisms, peer-to-peer and service discovery frameworks.» *Comput. Netw.*, 52, (2008), 2097–2128. URL http://dl.acm.org/citation.cfm?id=1383671.1384026.

[Mo09] MONTRESOR, A., y M. JELASITY. «PeerSim: A scalable P2P simulator.» En *Proceedings of the Ninth International Conference on Peer-to-Peer Computing*. 2009, P2P'09, 99–100. URL http://dx.doi.org/10.1109/P2P.2009.5284506.

[Na07] NAICKEN, S., B. LIVINGSTON, A. BASU, S. RODHETBHAI, I. WAKEMAN, y D. CHALMERS. «The state of peer-to-peer simulators and simulations.» *SIGCOMM Comput. Commun. Rev.*, 37, (2007), 95–98. URL http://doi.acm.org/10.1145/1232919.1232932.

[Or11] ORTIZ, S. «Is Peer-to-Peer on the Decline?» *Computer*, 44, (2011), 11–13.

[Zho09] OU, Z., E. HARJULA, y M. YLIANTTILA. «Effects of different churn models on the performance of structured peer-to-peer networks.» En *Proceedings of the 20th International Symposium on Personal, Indoor and Mobile Radio Communications*.

[Pla97] PLAXTON, G., R. RAJARAMAN, y A. RICHA. «Accessing nearby copies of replicated objects in a distributed environment.» En *Proceedings of the ninth annual ACM symposium on Parallel algorithms and architectures*. ACM, Nueva York, EUA, 1997, SPAA '97, 311–320. URL http://doi.acm.org/10.1145/258492.258523.

[Pou05] POUREBRAHIMI, B., K. BERTELS, G. M. KANDRU, y S. VASSILIADIS. «A survey of peer-to-peer networks.» En *Proceedings of the 16th Annual Workshop on Circuits, Systems and Signal Processing*. Veldhoven, Holanda, 2005, ProRisc 2005, 570–577.

[Pu09] PUJOL-AHULLÓ, J., y P. GARCÍA-LÓPEZ. «PlanetSim: An extensible simulation tool for peer-to-peer networks and services.» En *Proceedings of the Ninth International Conference on Peer-to-Peer Computing*. 2009, P2P'09, 85–86. URL http://dx.doi.org/10.1109/P2P.2009.5284504.

[Ra01] RATNASAMY, S., P. FRANCIS, S. SHENKER, y M. HANDLEY. «A scalable content-addressable network.» *SIGCOMM Comput. Commun. Rev.*, 31, (2001), 161–172. URL http://doi.acm.org/10.1145/964723.383072.

[Rh04] RHEA, S., D. GEELS, T. ROSCOE, y J. KUBIATOWICZ. «Handling churn in a DHT.» En *Proceedings of the annual conference on USENIX Annual Technical Conference*. USENIX Association, Berkeley, EUA, 2004, ATEC '04, 1–10. URL http://dl.acm.org/citation.cfm?id=1247415.1247425.

[Rh01] RHEA, S., C. WELLS, P. EATON, D. GEELS, B. ZHAO, H. WEATHERSPOON, y J. KUBIATOWICZ. «Maintenance-Free Global Data Storage.» *IEEE Internet Computing*, 5, (2001), 40–49. URL http://dl.acm.org/citation.cfm?id=613350.613636.

[Ro01] ROWSTRON, A., y P. DRUSCHEL. «Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems.» En *Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms Heidelberg*. Springer-Verlag, Londres, Reino Unido, 2001, Middleware '01, 329–350. URL http://dl.acm.org/citation.cfm?id=646591.697650.

[Ste06] STEINMETZ, R., y K. WEHRLE. *Peer-to-Peer Systems and Applications (Lecture Notes in Computer Science)*, tomo 3485. Springer-Verlag New York, Inc., Secaucus, EUA, 2005.

[Sto04] STOICA, I., D. ADKINS, S. ZHUANG, y S. SURANA. «Internet indirection infrastructure.» *IEEE/ACM Trans. Netw.*, 12, (2004), 205–218. URL http://dx.doi.org/10.1109/TNET.2004.826279.

[Sto01] STOICA, I., R. MORRIS, D. KARGER, M. F. KAASHOEK, y H. BALAKRISHNAN. «Chord: A scalable peer-to-peer lookup service for internet applications.» *SIGCOMM Comput. Commun. Rev.*, 31, (2001), 149–160. URL http://doi.acm.org/10.1145/964723.383071.

[Stu05-1] STUTZBACH, D., y R. REJAIE. «Capturing Accurate Snapshots of the Gnutella Network.» En *Proceedings of the Global Internet Symposium*. 2005, GI'05, 127–132. URL http://www.barsoom.org/papers/gi05.pdf.

[Stu06] —. «Understanding churn in peer-to-peer networks.» En *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*. ACM, Nueva York, EUA, 2006, IMC '06, 189–202. URL http://doi.acm.org/10.1145/1177080.1177105.

[Va02] VAHDAT, A., K. YOCUM, K. WALSH, P. MAHADEVAN, D. KOSTIC, J. CHASE, y D. BECKER. «Scalability and accuracy in a large-scale network emulator.» *SIGOPS Operating Systems Review*, 36, (2002), 271–284. URL http://doi.acm.org/10.1145/844128.844154.

[Yi01] YIANILOS, P. N., y S. SOBTI. «The Evolving Field of Distributed Storage.» *IEEE Internet Computing*, 5, (2001), 35–39. URL http://dl.acm.org/citation.cfm?id=613350.613635.

[Zha01] ZHAO, B. Y., J. D. KUBIATOWICZ, y A. D. JOSEPH. «Tapestry: An infrastructure for fault-tolerant wide-area location and routing.» *Inf. Téc. UCB/CSD-01-1141*, Berkeley, EUA, 2001.

[Zhi06] ZHIYU, L., Y. RUIFENG, L. ZHENHUA, L. HONGXING, y C. GUIHAI. «Survive under high churn in structured p2p systems: evaluation and strategy.» En *Proceedings of the International Conference on Computational Science*. 2006, ICCS '06, 404–411. URL http://cs.nju.edu.cn/~gchen/newpaper/UploadFiles/ICCS2006.pdf.

[Zhu01] ZHUANG, S., B. ZHAO, A. JOSEPH, R. KATZ, y J. KUBIATOWICZ. «Bayeux: an architecture for scalable and fault-tolerant wide-area data dissemination.» En *Proceedings of the 11th international workshop on Network and operating systems support for digital audio and video*. ACM, Nueva York, EUA, 2001, NOSSDAV '01, 11–20. URL http://doi.acm.org/10.1145/378344.378347.

Evaluación de la confiabilidad de índices P2P en presencia de alta transitoriedad

Idónea Comunicación de Resultados para obtener el grado de
MAESTRO EN CIENCIAS Y TECNOLOGÍAS DE LA INFORMACIÓN
2011

Dirigida por la Doctora
Elizabeth Pérez Cortés

Departamento de Ingeniería Eléctrica
Posgrado en Ciencias y Tecnologías de la Información
Universidad Autónoma Metropolitana - Iztapalapa

5 de octubre de 2011