UNIVERSIDAD AUTÓNOMA METROPOLITANA

Comparación de mecanismos de incentivos en sistemas par a par

Idónea comunicación de resultados presentada por
María Esther Sosa Rodríguez
Para obtener el grado de
Maestra en Ciencias
(Ciencias y Tecnologías de la Información)

Asesora:
Dra. Elizabeth Pérez Cortés

Jurado Calificador:

Presidente: Dr. Santiago Domínguez Domínguez	CINVESTAV - IPN
Secretario: Dra. Elizabeth Pérez Cortés	UAM-I
Vocal: Dr. Ricardo Marcelín Jiménez	UAM-I
Vocal: Dr. Manuel Aguilar Cornejo	UAM-I

27 de enero de 2012

Dedicatoria

Para

Emiliano y Ezequiel

con todo mi amor.

Agradecimientos

A la Universidad Autónoma Metropolitana por la oportunidad de realizar mis estudios de maestría, así como al Consejo Nacional de Ciencia y Tecnología (CONACyT) por el financiamiento brindado.

A la Dra. Elizabeth Pérez Cortés por haberme permitido trabajar bajo su asesoría y por el gran apoyo y conocimiento compartido, tanto para el desarrollo de este proyecto de investigación, como en el ámbito personal. Mil gracias por todo.

A los profesores Dr. Santiago Domínguez Domínguez, Dr. Ricardo Marcelín Jiménez y Dr. Manuel Aguilar Cornejo, por el tiempo que dedicaron a la revisión de este proyecto de investigación, así como los comentarios y sugerencias para mejorarlo.

A todos mis profesores por los conocimientos compartidos.

A mi esposo Ezequiel y a mi hijo Emiliano por el gran amor que me inspiran y el que me muestran cada día y por ser quienes me motivan para ser mejor.

A las familias Morales Martínez y Sosa Rodríguez quienes constituyen mi gran familia y de quienes he recibido apoyo en todo momento.

A mi madre por su comprensión, amor y su consejo siempre oportuno. A mi padre por el infinito amor y el gran cariño que siempre me mostró.

A mi suegra, Doña Lu, por su tiempo y por el esmero que ha puesto en el cuidado de mi hijo.

A todos y cada unos de mis amigos y compañeros de la maestría.

Por último pero no menos importante, mi infinito agradecimiento a Dios por la vida que cada día me permite disfrutar.

Resumen

Los sistemas distribuidos par a par (P2P, peer-to-peer en inglés) son utilizados para una diversidad de fines, tales como: comunicación y colaboración, cómputo distribuido, servicios de Internet, distribución de contenido y almacenamiento de información, entre otros, además de estar constituidos por nodos que comparten sus recursos para el buen funcionamiento del sistema. Estos recursos pueden ir desde espacio de almacenamiento, ancho de banda, procesamiento de datos, etc. Una característica importante de los nodos es que son autónomos, es decir, tienen la libertad de decidir qué, cuándo y cuánto cooperar con el sistema. Esta autonomía tiene un impacto directo en la disponibilidad de los servicios y recursos que el sistema P2P ofrece a sus usuarios (otros nodos) ya que existe la posibilidad de que los nodos se comporten de manera egoísta y busquen obtener beneficios sin aportar recursos al sistema, poniendo en riesgo la disponibilidad. Debido a lo anterior, se han propuesto mecanismos para incentivar la cooperación de los nodos. Los incentivos están basados en pagos monetarios, reciprocidad o reputación. Dentro de los mecanismos de incentivos basados en reputación se encuentra un subconjunto que aplica la Teoría de Juegos para analizar el comportamiento de los nodos.

La teoría de juegos permite estudiar la situación estratégica de los nodos (cooperar, no cooperar) bajo la hipótesis de que siempre actúan racionalmente y, en consecuencia, buscan el máximo beneficio. En un sistema par a par, los nodos no siempre son racionales y además su comportamiento puede cambiar conforme pasa el tiempo. Para estudiar la evolución del comportamiento de individuos en una población, se utilizan técnicas de cómputo evolutivo.

En este trabajo de investigación hacemos uso de la teoría de juegos para modelar la situación estratégica de los nodos de un sistema P2P. Definimos un juego llamado Juego de Duplicación, donde los pares deben decidir si almacenan o no un documento que solicita ser guardado en el sistema. Si no hay suficientes candidatos para almacenar, el documento se pierde pero si los hay el documento permanece y el sistema gana valor. Enseguida se diseñaron dos mecanismos de incentivos: en el primero se les paga a los nodos cuando cooperan con el sistema y se les cobra cuando consumen mientras que, en el segundo, se evalúa el comportamiento de los nodos en función de su actitud (cooperativa o no), permitiéndoles consumir en consecuencia. Posteriormente aplicamos algoritmos genéticos para observar cómo evoluciona el comportamiento de los nodos cuando se les somete a los mecanismos propuestos. Finalmente realizamos un conjunto de simulaciones que nos permitieron ver qué mecanismo

tiene mayor éxito motivando la cooperación de los nodos y analizamos la composición de las poblaciones finales, considerando tanto las estrategias de los nodos que las componen (comportamiento programado) como la ejecución de esas estrategias durante el experimento (comportamiento exhibido).

Contenido

Lista de Figuras **11**

Lista de Tablas **13**

1. Introducción **15**

2. Sistemas distribuidos P2P y la disponibilidad **17**
2.1. Sistemas distribuidos P2P 17
2.1.1. Características de los sistemas P2P 18
2.1.2. Retos en los sistemas distribuidos P2P 19
2.1.3. Clasificación de los sistemas P2P 19
2.2. Disponibilidad en los sistemas P2P 21
2.3. Duplicación 22
2.3.1. Duplicación simple 23
2.3.2. Duplicación por bloques 23
2.3.3. Duplicación por códigos de borrado 24
2.4. Discusión 25

3. Mecanismos de incentivos y teoría de juegos **27**
3.1. Mecanismos de incentivos 27
3.1.1. Características de un mecanismo de incentivos 27
3.1.2. Clasificación de los mecanismos de incentivos 28
3.2. Teoría de juegos 29
3.2.1. Teoría de juegos y sistemas P2P 31
3.3. Discusión 37

4. La evolución de las estrategias en los juegos iterados **39**
4.1. Algoritmos evolutivos 39
4.1.1. Generación de población inicial 40
4.1.2. Evaluación de la población 41
4.1.3. Selección de los individuos 41
4.1.4. Reproducción 42
4.2. Algoritmo genético 45

4.2.1. Aplicación de algoritmos genéticos: *Encontrar el máximo de la función* $f(x) = x^2$ . . . 45
4.3. El dilema del prisionero iterado de Axelrod . . . 47
4.4. Discusión . . . 49

5. Evolución de la cooperación para la duplicación utilizando algoritmos genéticos 51
5.1. Problema . . . 51
5.2. Objetivo general . . . 52
5.3. Metodología . . . 52
5.4. Diseño de mecanismos de compensación para motivar la cooperación de la duplicación . . . 53
5.4.1. Mecanismo de compensación económica . . . 53
5.4.2. Mecanismo de compensación por reconocimiento social . . . 53
5.5. Diseño del juego de duplicación . . . 54
5.5.1. Descripción del juego . . . 54
5.5.2. Elección de los nodos . . . 56
5.6. Aplicación de algoritmos genéticos a los mecanismos de compensación y al juego de duplicación . . . 58

6. Experimentación y análisis de resultados 61
6.1. Descripción general . . . 61
6.2. Parámetros . . . 62
6.2.1. Población . . . 62
6.2.2. Mecanismos de compensación . . . 63
6.2.3. Elección . . . 64
6.2.4. Documentos . . . 64
6.2.5. Costo por consultas . . . 64
6.2.6. Parámetros para el algoritmo genético . . . 64
6.3. Experimentación . . . 65
6.3.1. Mejor mecanismo de compensación: económica y de reconocimiento social 65
6.3.2. Composición de la población al final de las generaciones . . . 67

7. Conclusiones y trabajo futuro 73

Apéndices 75

A. Implementación del juego de duplicación, de los mecanismos de compensación y del algoritmo genético 77
A.1. Requerimientos . . . 77
A.1.1. Funcionales . . . 77
A.1.2. No funcionales . . . 78
A.2. Arquitectura . . . 78

A.2.1. Descripción general 78
A.2.2. Coordinador 79
A.2.3. Generador de población inicial 81
A.2.4. Juego de duplicación 82
A.2.5. Evaluación de la población 83
A.2.6. Selección de nodos 83
A.2.7. Reproducción 84
A.3. Plataforma de desarrollo 84

Referencias **85**

Lista de Figuras

2.1. Arquitectura de un Sistema Distribuido P2P 18
2.2. (A) P2P centralizados (B) P2P semicentralizados (C) P2P Puros (descentralizados) 21
2.3. Esquema de duplicación por códigos de borrado 24
2.4. La disponibilidad de los nodos en el sistema varía. 26

4.1. Diagrama de flujo de un algoritmo evolutivo 40
4.2. Mutación de un cromosoma 43
4.3. Diferentes técnicas de cruza: (A) un punto, (B)dos puntos, (C) uniforme. . . 44
4.4. Representación de un cromosoma: (a)binaria, (b)entera, (c)decimal. 45
4.5. Cromosomas para el dilema del prisionero iterado. 48

5.1. Algoritmo de ejecución para el juego de duplicación. 56
5.2. Técnicas de selección: (A) Secuencial, (B) Aleatoria, (C) Turno Rotatorio . . 57
5.3. Técnica para armar una estrategia (cromosoma). 59
5.4. Ejemplificación de la cruza para el juego de duplicación. 60

6.1. Promedio de documentos almacenados en cada generación, utilizando los dos mecanismos de compensación y elección (A) secuencial, (B) aleatoria y (C) turno rotatorio. 66
6.2. Aptitud promedio de los nodos en cada generación, utilizando los dos mecanismos de compensación y elección (A) secuencial, (B) aleatoria y (C) turno rotatorio. 68
6.3. Comportamieto programado y exhibido de los nodos cuando se utiliza el mecanismo de reconocimiento social. 71
6.4. Comportamieto programado y exhibido de los nodos cuando se utiliza el mecanismo de compensación económica. 72

A.1. Vista general del simulador. 79
A.2. Diagrama de secuencia de la ejecución del algoritmo genético. 80
A.3. Diagrama de secuencia del módulo de Evaluación de la Población. 82

Lista de Tablas

2.1. Disponibilidad de un sistema distribuido en función del tiempo que no está disponible en un año [10] 22

3.1. Dilema del prisionero representado en forma normal 31

4.1. Representación de posibles soluciones al problema, como cadenas binarias (cromosomas) 45
4.2. Evaluación de la población cuya función de aptitud es $f(x) = x^2$ 46
4.3. Muestra el valor esperado que obtiene cada individuo de la población 46
4.4. Parejas seleccionadas para generar los individuos de la siguiente generación. . 47
4.5. Población que se considerará para la siguiente generación. 47

5.1. Distintos valores de costo por consulta de documentos, en relación con el umbral de pago de los nodos. 53
5.2. Distintos porcentajes de consulta de documentos, en función de la calificación obtenida para cada nodo. 54

A.1. Datos de entrada requeridos para la ejecución del simulador. 80

Capítulo 1
Introducción

Hoy en día, los sistemas distribuidos P2P (peer-to-peer en inglés) son ampliamente utilizados para una diversidad de aplicaciones, tales como: comunicación y colaboración, cómputo distribuido, servicios de Internet, bases de datos, y distribución y almacenamiento de información, entre otros [5].

Dos características importantes de los sistemas distribuidos P2P son: 1) los nodos que forman parte del sistema son autónomos, es decir, no existe una entidad que los organice o controle y 2) los nodos pueden tomar el rol de cliente y servidor al mismo tiempo, es decir, pueden solicitar recursos a otros nodos al mismo tiempo que comparten los propios. BitTorrent, Napster, Tapestry y OceanStore, son algunos sistemas de este tipo.

La característica de autonomía de los nodos en los sistemas P2P, tiene un gran impacto en su funcionamiento y en la calidad de servicio que pueden otorgar a los usuarios, esto debido a que los nodos tienen la libertad de decidir qué, cuándo y cuánto cooperar con el sistema. Estudios realizados a distintos sistemas P2P [35], como Gnutella, Maze P2P y eDonkey, muestran que los nodos consumen más recursos de los que contribuyen al sistema, y que además solo una pequeña proporción de nodos distribuye la mayoría de los recursos.

La problemática antes descrita, ha motivado a investigadores a desarrollar mecanismos que motiven la cooperación de los nodos en un sistema P2P, aplicando para ello Teoría de juegos, herramienta que permite estudiar la situación estratégica de los nodos (cooperar, no cooperar) y que asume racionalidad absoluta en su comportamiento. Por otro lado, también se han realizado trabajos donde se hace uso de técnicas evolutivas que permiten estudiar la evolución del comportamiento de individuos que se encuentran en situaciones estratégicas, tal como el trabajo realizado por el Dr. Robert Axelrod [37], en el cual utiliza algoritmos géneticos para observar el comportamiento de los presos del juego del dilema del prisionero.

El presente trabajo de investigación tiene por objetivo, observar la evolución del comportamiento de los nodos de un sistema P2P, cuando reciben incentivos por su cooperación (duplicar), para lo cual modelamos la situación estratégica de los nodos(duplicar, no duplicar) y el sistema con un juego, y diseñamos mecanismos que compensen la cooperación de los nodos.

Para observar la evolución del comportamiento de los nodos, utilizamos algoritmos genéticos y adoptamos ideas del trabajo realizado en [37].

Para alcanzar los objetivos antes mencionados, se siguió la siguiente metodología:

1. Estudio de las características de los sistemas P2P.
2. Estudio de la Teoría de juegos, así como de los trabajos de investigación realizados en sistemas P2P en los cuales se aplicó.
3. Estudio de algoritmos genéticos.
4. Diseño de mecanismos que evalúen la cooperación de los nodos con respecto a la duplicación de recursos (documentos) en un sistema P2P.
5. Diseño de un juego que permita modelar la situación estrategia (duplicar o no duplicar) de los nodos y el sistema P2P, el cual considera los mecanismos antes mencionados.
6. Implementación del juego propuesto, aplicando los mecanismos propuestos y algoritmos genéticos.
7. Experimentación y análisis de los resultados obtenidos.

Como resultado de seguir la metodología antes descrita, se tienen las aportaciones siguientes:

- Juego de duplicación que modela la situación estratégica de los nodos y el sistema P2P.
- Dos mecanismos de compensación. El primero que se basa es un esquema de pagos monetarios y el segundo que se basa en la reputación de los nodos.
- Un simulador que permite aplicar algoritmos genéticos a una población (en el contexto de este trabajo de investigación la población son los nodos que constituyen el sistema P2P).
- Análisis del impacto al comportamiento de los nodos, cuando se encuentran bajo los mecanismo de compensación.

El documento que aquí presentamos, está organizado de la siguiente manera:

El capítulo 2 describe las característcas principales de los sistemas distribuidos P2P. Mecanismos de incetivos, Teoría de juegos y su aplicación en los sistemas P2P son abordados en el capítulo 3. Para la aplicación de algoritmos genéticos, se realizó un estudio de esta técnica, la cuál es descrita en el capítulo 4. Posteriormente en los capítulos 5 y 6, se describe la propuesta para este trabajo de investigación y la experimentación y análisis de resultados respectivamente. Por último en el capítulo 7 se presentan las conclusiones y el trabajo futuro.

Capítulo 2
Sistemas distribuidos P2P y la disponibilidad

Un sistema distribuido es un conjunto de computadoras (nodos) conectadas entre sí, que comparten recursos como: contenidos, espacio de almacenamiento, ancho de banda, entre otros. El Internet es un ejemplo de un sistema distribuido ya que es un conjunto de millones de computadoras interconectadas que comparten recursos entre millones de usuarios.

Dos de las arquitecturas de sistemas distribuidos [1] son: par a par (P2P por sus siglas en inglés) y cliente-servidor. En el primero no existe una única entidad que responda las solicitudes de los usuarios, sino que todos los nodos son capaces de responder las solicitudes que se lleven a cabo. En el segundo existe una única entidad, el servidor, quien responde a las solicitudes y los clientes quienes hacen las solicitudes de servicio. Para este trabajo de investigación nos enfocaremos en los sistemas distribuidos con arquitectura P2P.

2.1. Sistemas distribuidos P2P

Específicamente un sistema distribuido P2P es aquel en cual los nodos que lo integran tienen la misma jerarquía y los recursos que se comparten están descentralizados, es decir, no existe una autoridad central que esté dedicada a responder solicitudes de otros nodos, sino que todos tienen esta capacidad, además de la de hacer consultas [2] (fig. 2.1).

En los sistema distribuidos P2P, los nodos además de compartir recursos se encargan de controlarlos y administrarlos de manera transparente.

Una definición de los sistemas P2P podemos encontrarla en [2] y enuncia lo siguiente: « Son sistemas distribuidos que consisten de nodos capaces de auto-organizarse dentro de topologías de red con el propósito de compartir recursos tales como: contenido, ciclos de CPU, espacio de almacenamiento y ancho de banda, capaces de adaptarse a fallas y al reacomodo de los nodos mientras mantienen una aceptable conectividad y rendimiento, sin requerir de intermediarios o soporte de una autoridad o servidor centralizado. »

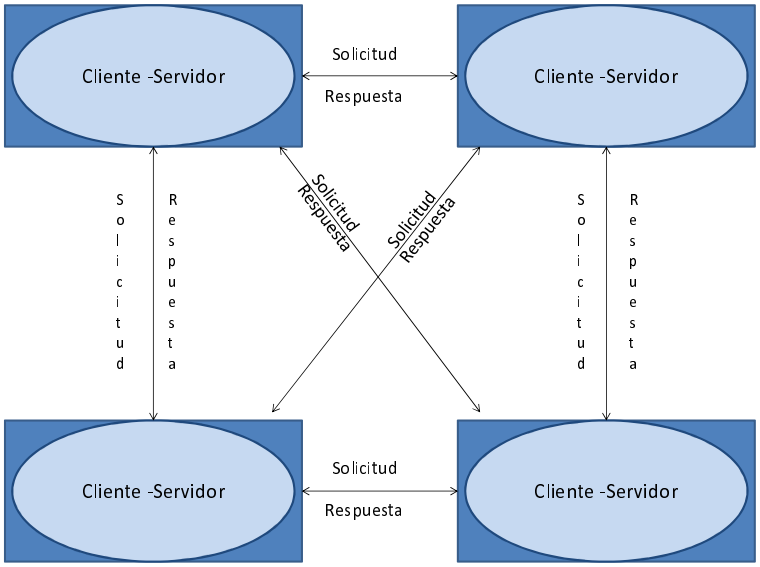

Figura 2.1: Arquitectura de un Sistema Distribuido P2P

Se han diseñado un gran número de sistemas P2P que buscan enfrentar los retos antes mencionados, como sistemas de: intercambio de archivos, publicación de contenido, almacenamiento de información, cómputo distribuido, etc. Napster, y OceanStore [3][4] son ejemplos de este tipo de sistemas.

En el resto de este capítulo describiremos las principales características, ventajas, desventajas y aplicaciones de los sistemas distribuidos P2P.

2.1.1. Características de los sistemas P2P

Algunas de las características principales de los sistemas P2P son las siguientes:

- Robustez: la información es duplicada en diferentes nodos del sistema, para que no se dependa de un servidor central y sea encontrada cuando es solicitada.
- Escalabilidad: el número de usuarios y recursos puede crecer en millones, lo cual no debe afectar el funcionamiento del sistema si no más bien mejorarlo.
- Seguridad: evitar a aquellos nodos cuyas acciones pueden afectar el funcionamiento del sistema, como por ejemplo aquellos que generan ataques de denegación de servicio.
- Tolerancia a fallas: el sistema es capaz de seguir ofreciendo el servicio aún y cuando se presente alguna falla en los componentes del sistema, sin que los usuarios perciban la falla.

- Arquitectura de Red: crear redes lógicas sobre las redes físicas que permitan un buen funcionamiento del sistema.

2.1.2. Retos en los sistemas distribuidos P2P

En el diseño y elaboración de sistemas distribuidos, debido a su naturaleza, es necesario tener en cuenta las siguientes consideraciones con respecto a los nodos que lo integran:

- Tienen una gran heterogeneidad.
- Son intermitentes, ya que pueden integrarse o dejar el sistema en cualquier momento, generando un alto nivel de transitoriedad[1].
- Se presentan retardos en las búsquedas y actualizaciones debido al tráfico que se genera en la red por las peticiones que se envían.
- Se presenta baja disponibilidad de los recursos en el sistema, debido a los cortes en la red y la salida de los nodos del sistema sin previo aviso.

2.1.3. Clasificación de los sistemas P2P

Los sistemas P2P, se pueden clasificar por:

- Tipo de aplicación [5].
- Grado de centralización [2] [5].
- Estructura de red [5].

Tipo de aplicación

- Comunicación y colaboración: facilitan la comunicación y la colaboración en tiempo real entre los nodos del sistema, como por ejemplo la mensajería instantánea (Yahoo, Messenger).
- Cómputo distribuido: permiten tomar ventaja de los nodos que están disponibles en el sistema, utilizando su poder de procesamiento. Por ejemplo en el cómputo paralelo se distribuyen cargas de trabajo pequeñas entre un determinado número de nodos para obtener un único resultado. En este tipo de sistemas es necesario contar con un coordinador central que es quien recibirá el resultado del grupo de nodos que están trabajando. Algunos sistemas de este tipo son GenomeAtHome y SetiAtHome [6] [7].

[1]Transitoriedad se define como: Característica de algo que no es definitivo, que no está destinado a perdurar mucho tiempo.

- Soporte a servicios de Internet: este tipo de sistemas ofrecen seguridad, mensajería, infraestructura, etc., al servicio de Internet, como por ejemplo el correo electrónico y los servicios de noticias.
- Sistemas de bases de datos: son sistemas distribuidos basados en P2P que contienen bases de datos. Sistemas de este tipo son: PIER, Piazza y Edutella.
- Distribución de contenido y almacenamiento de información: la mayor parte de los sistemas distribuidos P2P se encuentran dentro de esta categoría y están dedicados a la consulta e intercambio de datos entre los nodos del sistema, así como al almacenamiento distribuido de recursos. Se han diseñado sistemas sencillos para el intercambio de archivos como Napster, y también muy elaborados como OceanStore.

Grado de centralización

- Centralizados: cuentan con un servidor central en el cual se guarda información de los nodos y la ubicación de los recursos en el sistema. El servidor es el punto de enlace entre todos los nodos del sistema, lo cual facilita la búsqueda de recursos. Sin embargo, al ser estos servidores los únicos que contienen esta información, constantemente se generan retardos en las respuestas de las solicitudes realizadas por los nodos, convirtiéndose el servidor en la causa principal de falla para el sistema. Napster [3] es un sistema P2P centralizado.
- Semicentralizados: se tiene un grupo de nodos o supernodos que están interconectados y que manejan la información de los recursos compartidos, la cual es desconocida por el resto de los nodos. Estos supernodos no son servidores centrales por lo tanto no son puntos de falla como en el caso de los centralizados. En el caso de generarse una falla en un supernodo, el sistema es capaz de reemplazarlo con otro. BitTorrent [8] es un ejemplo de un sistema P2P semicentralizado.
- Puros: se caracterizan por no contar con algún servidor central, los nodos que conforman el sistema fungen como almacenes y usuarios, es decir, los nodos actúan como cliente-servidor. Freenet [9] es un sistema de este tipo. La figura 2.2 muestra las topologías de la clasificación por grado de centralización.

Tipo de red

- No estructurados: en estos sistemas los enlaces entre los nodos se forman de manera arbitraria y los recursos que contiene están relacionados con la red superpuesta que se forma con las conexiones entre los nodos. Para que un recurso sea encontrado es necesario localizar al nodo o nodos que lo contienen, para lo cual se lleva a cabo una inundación de mensajes en los nodos de la red. Este tipo de sistemas es muy utilizado cuando hay un grado alto de transitoriedad en los nodos. Sin embargo, no hay garantía

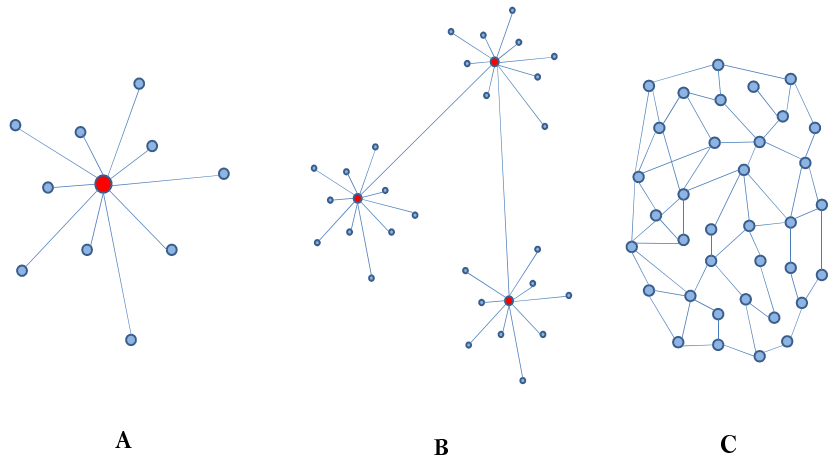

Figura 2.2: (A) P2P centralizados (B) P2P semicentralizados (C) P2P Puros (descentralizados)

de que los recursos sean localizados y constantemente se generan cuellos de botella debido a los retardos en las solicitudes y respuestas de los nodos. Algunos ejemplos son: Napster, Gnutella y Kazaa.

- Estructurados: para estos sistemas, la red que forman las conexiones entre los nodos es fuertemente controlada y los recursos son colocados en ubicaciones específicas. Estos sistemas se apoyan en el uso de tablas distribuidas que crean un mapeo entre los recursos y su ubicación, lo que hace que la búsqueda de los recursos sea efectiva, ya que el nodo que quiere localizar un recurso consultará la tabla distribuida y encaminará su petición hacia nodos específicos. Una desventaja en estos sistemas es que el mantenimiento de las tablas distribuidas es difícil debido a la transitoriedad de los nodos. Ejemplos de este tipo de sistemas son Chord, PAST, Tapestry.

En párrafos anteriores hemos descrito las características y los problemas que presentan los sistemas P2P, así como los tipos de sistemas que se han desarrollado. Sin embargo, no se ha mencionado una característica primordial en los sistemas distribuidos P2P: la disponibilidad. En la siguiente sección se describirá qué es la disponibilidad, así como los esfuerzos realizados para alcanzarla como objetivo dentro de los sistemas distribuidos P2P.

2.2. Disponibilidad en los sistemas P2P

La disponibilidad de un sistema puede define como: el número de minutos que no está disponible en un año [10]. La tabla 2.1 muestra los distintos grados y porcentajes de disponibilidad en sistemas distribuidos. El número de nueves que se muestra en la columna de

Sistema	No disponible (min.)	Disponibilidad	Grado de Disponibilidad
No administrado	50,000	90.00 %	1
Administrado	5,000	99.00 %	2
Bien administrado	500	99.90 %	3
Tolerante a fallas	50	90.99 %	4
Alta disponibilidad	5	99.999 %	5
Muy alta disponibilidad	0.5	99.9999 %	6
Ultra alta disponibilidad	0.05	99.99999 %	7

Tabla 2.1: Disponibilidad de un sistema distribuido en función del tiempo que no está disponible en un año [10]

disponibilidad nos dice cuál es el grado de disponibilidad que tiene un sistema, así bien un sistema con grado 5 indica que tiene alta disponibilidad, es decir, que está un 99.999 % disponible en un año.

Sin importar qué tipo de criterio utilicemos para establecer cómo medir la disponibilidad de un sistema distribuido P2P, para que un sistema alcance un nivel determinado de disponibilidad es necesario que los recursos que almacena estén disponibles [11], y para ello se requiere que los nodos en donde se encuentran almacenados también lo estén.

Para alcanzar la disponibilidad de los recursos que se encuentran almacenados en un sistema se utiliza la duplicación, que permite manter un determinado nivel de disponibilidad de los recursos, creando un determinado número de copias y distribuyéndolas en el sistema.

2.3. Duplicación

La duplicación es la habilidad que tiene el sistema para mantener copias de los recursos, específicamente contenido, de acuerdo a las necesidades del sistema.

Al implementar mecanismos de duplicación en un sistema P2P, se busca mantener un nivel apropiado tanto de disponibilidad como de rendimiento, permitiendo obtener los siguientes beneficios [11] [12]:

- Bajas latencias en las consultas ya que los recursos que se soliciten podrán localizarse fácil y rápidamente.
- Buen manejo del balance de carga ya que los recursos no estarán concentrados en una única ubicación sino de acuerdo a las necesidades del sistema.
- Aceptable fiabilidad, ya que los recursos serán encontrados en función del número de copias generadas y de la ubicación de las mismas.

Se han diseñado técnicas de duplicación de datos que han sido probadas y evaluadas [12] y que han generado buenos resultados, como los que se describen a continuación.

2.3.1. Duplicación simple

En la duplicación simple se hacen copias completas de los archivos en un número determinado de nodos del sistema según se requiera. Sin embargo aunque esta técnica es sencilla, es altamente costosa ya que puede ser complicado el manejo del espacio y el tiempo, especialmente en sistemas que deben soportar almacenamiento de archivos de gran tamaño.

En [13] se propone la siguiente ecuación para obtener la disponibilidad de un objeto en función de las copias distribuidas en un determinado número de nodos en el sistema:

$$A = 1 - (1 - a)^{k_s} \tag{2.1}$$

donde:

- k_s es el factor de redundancia del archivo, es decir, el número de copias que se van a distribuir, una en cada nodo.
- A es la disponibilidad del archivo.
- a es la disponibilidad promedio de los nodos en el sistema.

Despejando el factor de redundancia, se obtiene la siguiente ecuación:

$$k_s = \frac{\log(1 - A)}{log(1 - a)} \tag{2.2}$$

la cual nos permite saber cuantas copias del objeto se necesitan distribuir en el sistema para alcanzar un determinado nivel de disponibilidad.

2.3.2. Duplicación por bloques

En este enfoque se divide el archivo en m bloques de tamaño fijo, para después hacer n copias de cada bloque, las cuales se almacenan en diferentes nodos. La división del archivo en bloques ayuda a tener un mejor manejo del espacio con el que se cuenta para almacenar. El problema de esta técnica es que cuando se requiere reconstruir el archivo, es necesario que al menos una copia de cada bloque esté disponible, de lo contrario el archivo no se podrá reconstruir y como consecuencia no estará disponible.

La ecuación propuesta en [13] para conocer la disponibilidad de un objeto utilizando duplicación por bloques es la siguiente:

$$A = (1 - (1 - a)^{k_s})^m \tag{2.3}$$

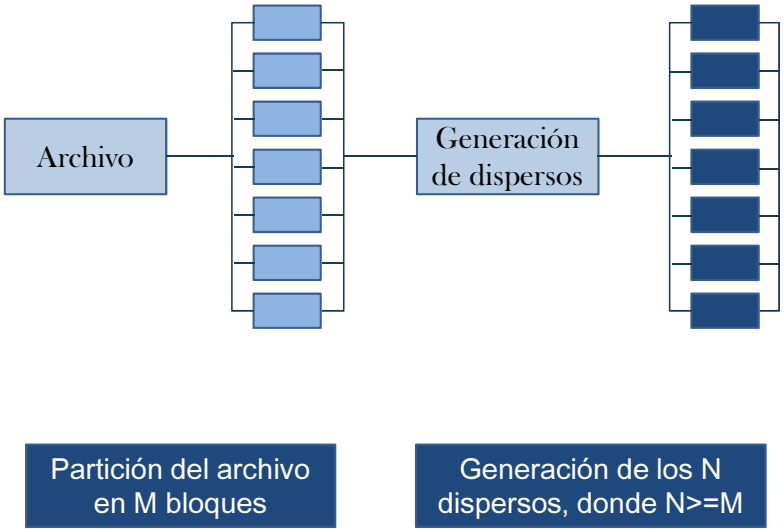

Figura 2.3: Esquema de duplicación por códigos de borrado

donde m es el número de bloques en que se descompone un objeto. Despejando el factor de redundancia se tiene:

$$k_s = \frac{\log(1 - A^{1/m})}{log(1 - a)} \tag{2.4}$$

la cual nos dice el número de copias de los m bloques que se distribuyen para obtener determinado nivel de disponibilidad

2.3.3. Duplicación por códigos de borrado

Los esquemas de código de borrado están caracterizados por su habilidad de proveer redundancia y por consiguiente resistencia a fallas, sin la sobrecarga de la duplicación simple. Para llevar a cabo este tipo de duplicación es necesario dividir un bloque de datos en m fragmentos todos del mismo tamaño, produciendo a partir de estos, n fragmentos recodificados, donde $n \geq m$. Después sólo es necesario obtener m cualesquiera de los n fragmentos producidos para reconstruir un bloque [14][15]. Estos esquemas también son conocidos como Códigos de Borrado m de n (fig. 2.3).

Para la duplicación por códigos de borrado, la ecuación de disponibilidad [13] es la siguiente:

$$A = \sum_{i=m}^{mk_I} \binom{mk_I}{i} a^i (1-a)^{mk_I - i} \tag{2.5}$$

donde k_I, el factor de estiramiento, es el cociente entre la cantidad de datos producidos después de llevar a cabo la partición y recodificación, entre la cantidad de datos originales.

Despejando el factor de estiramiento para el objeto, se tiene la siguiente ecuación:

$$k_s = \left(\frac{\sigma(A)\sqrt{\frac{a(1-a)}{b}} + \sqrt{\frac{\sigma(A)^2 a(1-a)}{b} + 4a}}{2a} \right)^2 \tag{2.6}$$

donde $\sigma(A)$ es la desviación estándar en una distribución de probabilidad normal dado un nivel de disponibilidad requerido. En [13] se detalla el valor que toma $\sigma(A)$.

2.4. Discusión

Los esfuerzos realizados para contar con buenas técnicas de duplicación, permiten que los sistemas distribuidos cuenten con un mejor nivel de disponibilidad de los recursos. Sin embargo, en los sistemas P2P sigue siendo un reto saber ¿qué técnica es la que conviene usar?, ¿dónde se guardarán las copias que se generan?, ¿cuántas copias generar?, etc., para mantener los recursos disponibles a los usuarios del sistema, y más importante aún considerar que la disponibilidad de estos recursos depende casi en su totalidad de la disponibilidad que presentan los nodos que los almacenan (fig. 2.4).

Los sistemas distribuidos P2P, como ya mencionamos, están constituidos por nodos autónomos, es decir, que tienen la capacidad de decidir en qué momento entran o dejan el sistema y con cuántos recursos contribuyen al mismo, y si nos enfocamos en un sistema de almacenamiento distribuido P2P, específicamente deciden cuánto espacio de almacenamiento compartir.

Si analizamos el comportamiento de los nodos (usuarios), estos van a actuar, la mayoría de la veces, de acuerdo a lo que más les conviene. Por ejemplo en un sistema distribuido del cual se puede descargar música, si un nodo no tiene restricción alguna para hacer las descargas que desee, entonces hará uso del sistema sin contribuirle nada, y si consideramos que la mayoría de los nodos se comportarán de la misma forma, entonces sólo un grupo pequeño de nodos estará contribuyendo con recursos al sistema.

La situación anterior ha llevado a los investigadores a pensar en aplicar modelos económicos que permitan implementar mecanismos en los cuales los nodos del sistema sean incentivados a cooperar con recursos [16]. Algunos de estos modelos ecónomicos se enfocan en utilizar el concepto de Teoría de Juegos (TJ) para implementar dichos mecanismos. La TJ analiza

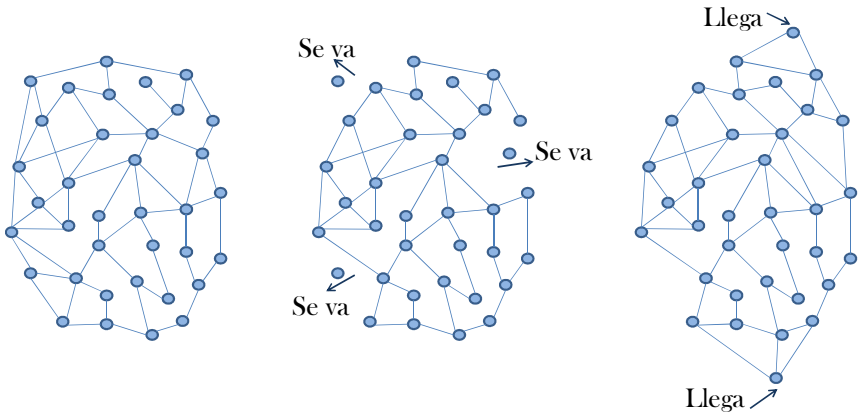

Figura 2.4: La disponibilidad de los nodos en el sistema varía.

problemas en los cuales múltiples personas llevan a cabo toma de decisiones para obtener un beneficio personal.[16]

La Teoría de juegos es una herramienta que puede aplicarse a los sistemas P2P, ya que considera la situación de toma de decisiones de múltiples individuos como un juego, a los individuos como jugadores, las decisiones que se toman como estrategias y el beneficio que obtienen de las decisiones tomadas como la utilidad. Si trasladamos estos conceptos a los sistemas distribuidos P2P, podemos decir que los nodos son los jugadores en el juego, las estrategias son: cooperar o no cooperar con el sistema, y la utilidad de los nodos es el beneficio que reciben por la estrategia elegida [22].

En el siguiente capítulo describiremos de manera breve la clasificación y características de los mecanismos de incentivos, así como los conceptos básicos de Teoría de juegos que nos serán útiles para la descripción del trabajo relacionado.

Capítulo 3
Mecanismos de incentivos y teoría de juegos

Para buscar una solución al problema de falta de cooperación que muestran los nodos de un sistema distribuido P2P, se han desarrollado diferentes mecanismos de incentivos que buscan motivar a los nodos a contribuir con el sistema. Algunos de estos mecanismos utilizan Teoría de juegos para modelar la situación estratégica de los nodos (cooperar, no cooperar), buscando fomentar la cooperación.

En la primera parte de este capítulo describimos las características de los mecanismos de incentivos, en la segunda parte describiremos los conceptos básicos de Teoría de juegos necesarios para describir el trabajo relacionado.

3.1. Mecanismos de incentivos

El estudio realizado en *"A Review of Incentive Mechanisms in Peer-to-Peer Systems "* [26] sugiere, que un mecanismo de incentivos, debe cumplir las características que describiremos a continuación.

3.1.1. Características de un mecanismo de incentivos

1. El mecanismo debe ser capaz de administrarse así mismo, es decir, no debe requerir de una entidad que monitoreé el comportamiento de los nodos.
2. El mecanismo debe ser aplicable a cualquier sistema P2P, sin importar el tipo de servicio que éste ofrezca.
3. El mecanismo debe incentivar a los nodos a contribuir con el sistema, para así mejorar la disponibilidad del mismo.
4. El mecanismo debe detectar a los nodos que se rehusan a cooperar, para de algún modo penalizar la falta de cooperación.
5. El mecanismo debe adaptarse al sistema P2P, sin afectar su funcionamiento.

3.1.2. Clasificación de los mecanismos de incentivos

Los mecanismos de incentivos son clasificados de la siguiente forma:

- Esquemas de pagos monetarios.
- Esquemas de contribución fija.
- Esquemas basados en reciprocidad.

Esquemas de pagos monetarios

En este tipo de incentivos, se consideran tres entidades importantes: 1) el nodo consumidor, 2) el intermediario y 3) el nodo comerciante. El primero es quien desea consumir un servicio del sistema, el segundo es el encargado de realizar las transacciones monetarias entre el consumidor y el comerciante y el tercero es quien ofrece el servicio. Se utilizan básicamente dos modelos para su implementación: 1) basado en token y 2) basado en cuenta.

En el primer modelo, el nodo consumidor verifica el servicio que un nodo comerciante le ofrece, si éste le es de interés, entonces el consumidor compra tokens al intermediario, con lo cual lleva a cabo el consumo del servicio. En el segundo modelo, el nodo consumidor verifica el servicio que un nodo comerciante le ofrece y si está interesado, informa al intermediario, con quien tiene una cuenta monetaria, que desea consumir el servicio que ofrece el comerciante. El intermediario notifica al comerciante que sus servicios son solicitados y éste a su vez, los provee al consumidor. Una vez que el consumidor recibió el servicio, confirma al intermediario y éste a su vez le paga al comerciante.

Estos dos modelos de esquemas de pagos monetarios: basado en token y basado en cuenta, han sido implementados en PPay [27] y PayPal [28] respectivamente.

Esquemas de contribución fija

En este esquema, los nodos que quieren unirse al sistema, deben de compartir un mínimo de archivos y deben poner a disposición un mínimo de ancho de banda del que tienen disponible. Direct Connect [29], aplica este esquema de incentivos.

Además es requisito que exista una autoridad central que monitoreé las contribuciones que realizan los nodos, por lo cual no es posible su aplicación en los sistemas P2P y limita su aplicación a sistemas dedicados a un único servicio.

Otra característica importante es que, debido a que los nodos sólo necesitan contribuir con un mínimo de sus recursos, estos no tendrán interés alguno por incrementar su contribución.

Esquemas basados en reciprocidad

Estos esquemas se basan en las contribuciones que hacen los nodos al sistema, es decir, los nodos podrán consumir servicios en función de su contribución. Este tipo de esquemas se divide en dos tipos: 1) esquemas basados en reciprocidad en tiempo real y 2) esquemas basados en reciprocidad en tiempo no real.

En el primero, los nodos que interactúan en una transacción se evalúan entre sí, en el momento en que se está llevando a cabo, motivando a los nodos a poner disponibles sus recursos en el momento en que ellos están consumiendo recursos de otros nodos. En BitTorrent [8] se aplica este esquema de incentivos [26].

En el esquema basado en reciprocidad en tiempo no real, se evalúa la reputación que han ganado los nodos en interacciones pasadas, lo cual es útil para predecir el comportamiento que pueden presentar en el futuro. Los siguientes trabajos de investigación utilizan este esquema de incentivos [26] el algoritmo EigenTrust [30], y los sistemas PowerTrust [31] y PeerTrust [32] . En [33], [21] y [34], también se utiliza este tipo de esquemas, pero además se hace uso de Teoría de Juegos, la cual es de nuestro interés para este trabajo de investigación.

3.2. Teoría de juegos

La teoría de juegos tuvo sus inicios en 1944 con John von Neumann y Oskar Morgenstern [17], y se define como: el estudio de problemas de decisión multi-personales [18]. Inicialmente la finalidad de la Teoría de Juegos fue encontrar una solución teórica a los problemas de incertidumbre en los juegos de azar con respecto a las decisiones que se toman dentro de estos juegos, pero con el tiempo este enfoque se fue adoptando dentro de la Economía, la Biología y la Computación entre otras.

Para que un problema pueda modelarse como un juego, es necesario contar con los siguientes elementos:

- Involucrados en el problema: Jugadores
- Conjunto de opciones con que cuenta cada jugador: Estrategias
- Resultados obtenidos al final de la elección de estrategias: Utilidad o Beneficio, el cual está determinado por la combinación de estrategias elegidas por los jugadores.

Es importante hacer énfasis en que se asume que los jugadores involucrados en el juego se comportan de manera racional, es decir, buscan maximizar su propio beneficio sin considerar el beneficio colectivo.

Dentro de la Teoría de juegos, se hace una clasificación de los tipos de juego de la siguiente manera:

- Juegos estáticos con información completa
- Juegos dinámicos con información incompleta
- Juegos dinámicos con información incompleta
- Juegos estáticos con información incompleta

Un juego estático se refiere a que los jugadores toman de manera simultánea las estrategias que van a utilizar en el juego, es decir, ningún jugador sabe cuales son las estrategias que elegirán los otros jugadores. En los juegos dinámicos, los jugadores toman sus decisiones considerando lo que han jugado los otros jugadores en una tirada anterior.

Cuando un juego tiene información completa, se refiere a que la función de utilidad que determina las ganancias de cada jugador de acuerdo a las decisiones que se tomaron, es de dominio público. Si el juego es con información incompleta, al menos un jugador no está seguro de las funciones de utilidad de los otros jugadores.

Las estrategias que tiene cada jugador en un juego, pueden ser de dos tipos: puras y mixtas. La estrategias puras, son aquellas en donde sólo una de ellas puede ser elegida, es decir, no hay elección al azar [19]. Si el conjunto general de estrategias tiene n subconjuntos, entonces se asigna el valor de 0 a n-1 de ellos y al n-ésimo subconjunto se le asigna el valor de 1. En las estrategias mixtas, se asigna una distribución de probabilidad fija sobre el conjunto de estrategias con que cuenta un jugador.

La finalidad de la teoría de juegos, es encontrar el equilibrio de Nash [18] [19] para el juego. Este concepto fue introducido en 1950 por el Dr. John Forbes Nash en su tesis de doctorado, en la cual prueba que los juegos finitos de forma estratégica tienen al menos un equilibrio de Nash [18]. Se dice que un equilibrio de Nash en un juego es alcanzado cuando los jugadores racionales que participan en él, eligen una estrategia que será la mejor respuesta a las estrategias elegidas por los otros jugadores, y una vez elegidas ningún jugador obtendrá un beneficio mayor por cambiar de estrategia [18].

Uno de los juegos más representativos en la Teoría de Juegos, es el Dilema del prisionero [18], en donde 2 personas son arrestadas por ser sospechosas de cometer un delito. La policía no tiene evidencia suficiente para condenarlos a menos que ellos confiesen. Como ningún delicuente quiere confesar, entonces son encerrados en celdas separadas explicándoles lo siguiente:

- Si ninguno confiesa (cooperar), ambos serán condenados a estar un mes en la cárcel.
- Si ambos confiesan (traicionar), serán condenados a seis meses en la cárcel.
- Si uno confiesa y el otro no, el que confiesa es puesto en libertad y el que no es sentenciado a nueve meses en la cárcel.

	confesar	no confesar
confesar	-6,-6	0, -9
no confesar	-9, 0	-1, -1

Tabla 3.1: Dilema del prisionero representado en forma normal

Como se puede observar, al estar los dos sospechosos en celdas separadas, ninguno tendrá conocimiento de la decisión tomada por su cómplice (aunque no necesariamente tomen sus decisiones de manera simultánea), pero sí tienen el conocimiento de las consecuencias que obtendrán de las decisiones que se tomen. El juego antes descrito se muestra representado en forma normal, en la tabla 3.1.

En el juego del dilema del prisionero, se observa que si los dos presos eligen confesar entonces se obtiene un equilibrio de Nash pues ninguno tendrá incentivos para cambiar su decisión, y si alguno de ellos decidiera no confesar, corre el riesgo de pasar más tiempo en prisión y que su cómplice quede en libertad.

Recientemente se ha aplicado la Teoría de Juegos a los sistemas P2P con la finalidad de observar el comportamiento de los nodos que participan en el sistema, llevando a cabo algunos trabajos de investigación de los cuáles daremos una breve descripción en la siguiente sección.

3.2.1. Teoría de juegos y sistemas P2P

La teoría de juegos puede bien aplicarse a los sistemas P2P, ya que considera la situación de toma de decisiones de múltiples individuos como un juego, a los individuos como jugadores, las decisiones que se toman como estrategias y el beneficio que obtienen de las decisiones tomadas como la utilidad. Si trasladamos estos conceptos a los sistemas distribuidos P2P, podemos decir que los nodos son los jugadores en el juego, las estrategias son: cooperar o no cooperar con el sistema, y la utilidad de los nodos es el beneficio que reciben por la estrategia tomada. A continuación describiremos los trabajos encontrados

Un marco teórico para el juego de incentivos en los sistemas P2P. *(A Game Theoretic Framework for Incentives in P2P Systems)*

En [21], el objetivo es implementar un modelo de incentivos económicos (dólares) aplicable a los sistemas distribuidos P2P, para encontrar un equilibrio entre los nodos participantes, los cuales obtendrán un beneficio o utilidad proporcional a los recursos con que contribuyen (disco duro, ancho de banda). El juego propuesto es un juego dinámico pues los nodos participantes se basan en las estrategias elegidas por los otros nodos en una iteración anterior.

Los componentes del juego son:

- Jugadores: nodos en el sistema
- Estrategias: cuánto contribuirán (recursos) al sistema
- Utilidad: minimizar costos por unirse al sistema

Para observar el costo de unirse al sistema, en [21] se plantea la siguiente ecuación:

$$u_i = -d_i + p(d_i) \sum_{i=1}^{n} b_{ij} d_j, \ \textit{donde } b_{ii} = 0 \tag{3.1}$$

donde el significado de las variables es el siguiente:

- d_i : contribución de recursos del nodo i
- $p(d_i)$: probabilidad de que una solicitud hecha por el nodo i sea aceptada
- b_{ij} : beneficio total que el nodo i obtiene del nodo j.
- d_j Contribución de recursos del nodo j

Para este trabajo no se considera alguna técnica de duplicación, pues se enfoca en los costos que implica a un nodo unirse al sistema. También se establece un valor crítico para el beneficio que pueden obtener los nodos, b_c. Si el beneficio que obtienen los nodos es menor que b_c, entonces al nodo no le conviene unirse al sistema.

La interacción entre los jugadores consta de dos acciones:

- Los nodos eligen una estrategia de contribución, la cual consiste para fines de este trabajo en decidir la cantidad de disco duro que contribuirán al sistema
- Cuando un nodo que ya efectuó su contribución requiere consumir algún recurso en el sistema, el servicio que se le entregará estará basado en la contribución que el nodo hizo al sistema

En el juego se busca encontrar un equilibrio en donde los nodos son vistos como:

- Homogéneos: todos los nodos obtienen el mismo beneficio
- Heterogéneos: los nodos obtienen diferente beneficio.

En el primer caso el juego es resuelto encontrando el equilibrio de Nash para un par de nodos homogéneos utilizando un modelo analítico, con el cual encuentran el valor de la contribución que les dará un mejor beneficio a los nodos y que se muestra a continuación.

$$d^* = \left(\frac{b}{2} - 1\right) \pm \left(\left(\frac{b}{2} - 1\right)^2 - 1\right)^{\frac{1}{2}} \tag{3.2}$$

Para resolver el sistema con nodos heterogéneos se describe un modelo de aprendizaje en el cual de manera iterativa los nodos eligen sus estrategias de contribución con base en las estrategias elegidas por los otros nodos en una iteración anterior. Al igual que en el sistema homogéneo, se llega a un equilibrio de Nash, esto como consecuencia de la elección iterativa de estrategias, pues en cada iteración los nodos modificarán sus estrategias de contribución y entonces, en un determinado momento, habrán elegido su mejor opción de contribución y no desearán modificarla.

El almacenamiento egoísta en Sistemas Distribuidos: un análisis de teoría de juegos. *(Selfish Caching in Distributed Systems: A Game-Theoretic Analysis)*

En [22], el objetivo es minimizar el precio de la anarquía, es decir, el costo que implica que un sistema P2P carezca de un coordinador y que los nodos que lo integran actúen de manera interesada cuando deciden duplicar o no un objeto. El juego es modelado como un juego estático y no cooperativo ya que los nodos no consideran las decisiones de los otros nodos en el sistema.

Los componentes del juego son:

- Jugadores: nodos en el sistema
- Estrategias: duplicar, no duplicar
- Utilidad: minimizar costos locales de almacenamiento

Se consideran dos tipos de juego, el juego básico donde no es considerado un incentivo para almacenar objetos, y el juego de pago en donde los nodos son incentivados a almacenar.

Las ecuaciones para cada juego son las siguientes:

$$C_i(S) = \alpha_{ij} S_i + w_{ij} d_{il(i,j)}(1 - S_i) \tag{3.3}$$

$$C_i(S) = \alpha_{ij} I_i + w_{ij} d_{il(i,j)}(1 - I_i) + b_i I_{vi} - R_i I_i \tag{3.4}$$

Cada una de las variables en las ecuaciones 3.3 y 3.4 tiene el siguiente significado:

- $i \in$: conjunto de nodos
- $j \in$: conjunto de objetos
- $d_{il(i,j)}$: es la distancia entre los servidores i, l
- w_{ij} : es la demanda del servidor i por el objeto j
- α_{ij} : costo de almacenar el objeto j en el servidor i
- S_i : opción elegida por el servidor (duplicar, no duplicar)
- v_i : representa al jugador que realiza una oferta al servidor i
- b_i : es el valor de la oferta propuesta al servidor i, donde $b \geq 0$
- I_i : indica si i replica o no el objeto , tomando el valor de 1 ó 0 respectivamente
- R_i : indica la cantidad total de la suma de las ofertas que recibe el nodo i

La técnica utilizada para duplicar los objetos es duplicación simple y los servidores no tienen establecidos límites en el espacio de almacenamiento.

La interacción entre los jugadores se da de la siguiente manera: para el juego básico los nodos reciben una solicitud para que dupliquen un objeto, en seguida calculan el costo que implica colocar el objeto, y la distancia entre el nodo solicitante y el nodo a quien se hizo la solicitud. Si el costo por colocar un objeto es mayor que la distancia entre los nodos, entonces optará por no duplicar, en caso contrario duplicará. Para el juego de pago, además se tomará en cuenta la oferta de pago que hace el nodo solicitante.

Resuelven el juego encontrando el equilibrio de Nash utilizando un método analítico para cada objeto, y de la unión de todos los equilibrios de Nash, se obtiene un equilibrio de Nash para un juego de múltiples objetos [22].

Duplicación egoísta distribuida. *(Distributed Selfish Replication (DSR))*

En [23], el objetivo es mejorar la utilidad global con estrategias de duplicación de equilibrio, con respecto a la utilidad obtenida utilizando estrategias ávidas. Esto último significa que los nodos no tendrán en cuenta las estrategias que elijan el resto de los nodos. El juego es modelado como un juego dinámico, no cooperativo y de suma-no cero [19] (la ganancia de un jugador no necesariamente corresponde con la pérdida de otro).

Los componentes del juego son:

- Jugadores: nodos en el sistema
- Estrategias: selección de conjunto de objetos a duplicar

- Utilidad: minimizar costos locales por duplicar

Para observar el costo de duplicación se plantea la siguiente ecuación:

$$G_{ij}(P) = \sum_{o_i \in P_j} r_{ij}(t_s - t_l) + \sum_{o_i \notin P_j, o_i \in p_{-j}} r_{ij}(t_s - t_r) \tag{3.5}$$

Para esta ecuación las variables representan lo siguiente:

- G_{ij} : ganancia del nodo j por replicar un objeto i en el paso k, $k \in \{0, 1\}$
- t_l, t_r, t_s : costos por acceder a un objeto de manera local, remota y al servidor de origen respectivamente.
- r_{ij} : indica la frecuencia con la que el objeto i es solicitado por el nodo j

Se utiliza el modelo de duplicación simple para las copias de los objetos.

La interacción entre los nodos es la siguiente: como primer paso los nodos son ordenados de manera creciente y se les asigna un turno de acuerdo a ese orden. La toma de decisiones no es simultánea, ya que la elección de las estrategias se realiza de acuerdo al turno asignado. Así, un nodo al que le toca su turno de jugar, elegirá sus estrategias considerando las estrategias de los nodos que eligieron en turnos menores a él. Las decisiones de los nodos con turno mayor al nodo actual, no es posible considerarlas pues aún no han elegido sus estrategias.

Los autores resuelven el juego encontrando el equilibrio de Nash, implementando un algoritmo de búsqueda local de dos pasos (TSLS por sus siglas en inglés), con el cual se obtienen las colocaciones de los objetos para cada uno de los nodos [23]. En el paso 0 o paso de inicialización los nodos eligen sus estrategias de duplicación de manera ávida, y en el paso 1 o paso de mejora, los conjuntos de estrategias de colocación generados en cada nodo en el paso 0 son observados y analizados por cada uno de los nodos, para así realizar la mejora de sus propias estrategias. Para realizar la mejora de sus estrategias, consideran el turno que les fue asignado. Es importante mencionar que si las estrategias generadas en el paso 0 maximizan las ganancias de un nodo, entonces es posible conservarlas.

El algoritmo se ejecuta de manera iterativa y en determinado momento los nodos habrán obtenido las estrategias que les redundan en un mayor beneficio, llegando así al equilibrio de Nash.

Un equilibrio de Nash puro basado en Teoría de juegos para la duplicación de datos entre varios servidores. *(A Pure Nash Equilibrium based Game Theoretical Method for Data Replication across Multiple Servers)*

En [24], el objetivo es implementar un algoritmo que lleva por nombre NCOR para resolver el problema de la duplicación y así minimizar los costos de transferencia de los recursos

en la Web. El juego para el sistema es modelado como un juego estático, el cual es utilizado como mecanismo generador de copias.

Los componentes del juego son:

- Jugadores: nodos en el sistema
- Estrategias: duplicar, no duplicar
- Utilidad: minimizar los costos de transferencia de los objetos, debido a las lecturas y escrituras de los mismos.

Para observar el costo que implica la transferencia de objetos se plantea la siguiente ecuación:

$$X = \sum_{i=1}^{M} \sum_{k=1}^{N} \left[(1 - X_{ik}) \left[r_k^i o_k min \left\{ c(i,j) | X_{ik} = 1 \right\} + w_k^i o_k c(i, P_k) \right] + X_k^i (\sum_{x=1}^{M} w_k^x) o_k c(i, P_k) \right] \tag{3.6}$$

donde las variables son descritas a continuación:

- M : nodos heterogéneos en el sistema
- N : objetos en el sistema
- $c(i,j)$: enlace entre i y j
- r_k^i, w_k^i : lecturas y escrituras respectivamente del objeto k realizadas por el nodo i
- P_k : servidor que almacena el original del objeto k
- NN_k^i : nodo vecino del nodo i que almacena el objeto k
- X_k^i : indica si el objeto fue almacenado: 1 si se duplica y 0 en otro caso

Se utiliza el modelo de duplicación simple para generar copias de los objetos.

El problema de la duplicación es modelado como una matriz, en la cual se busca encontrar la asignación de 0"s y 1"s que minimicen el costo total de transferencia de los objetos de acuerdo a las escrituras y lecturas que se realizan sobre ellos.

No hay una interacción directa entre los nodos, ya que no toman en cuenta las decisiones de duplicación que toman los otros nodos para seleccionar las estrategias propias, más bien sus decisiones están basadas en las características de los objetos, como son: tamaño, ubicación y frecuencia de lecturas y escrituras. Resuelven el juego encontrando el equilibrio de Nash por medio de un algoritmo en el cual los nodos eligen sus estrategias de duplicación, y

con base en estas verifican la utilidad que obtienen y actualizan su espacio de almacenamiento.

NCOR comienza su ejecución con un esquema de duplicación inicial, el cual converge a un equilibrio de Nash ya que cada agente o nodo modifica sus estrategias con el fin de minimizar sus costos de manera iterativa. Un equilibrio de Nash existe en NCOR cuando sobre el conjunto de N objetos, todos los nodos maximizan su beneficio, minimizando sus costos de comunicación.

Los resultados en NCOR son comparados con otros algoritmos que no utilizan el concepto de Teoría de Juegos [24], obteniendo el primero mejores resultados tanto en costos de comunicación, como en el número de copias que logra colocar.

3.3. Discusión

Los trabajos antes descritos proponen técnicas que permiten maximizar el beneficio o utilidad de los integrantes de un sistema P2P ya sea por minimizar costos de almacenamiento u otorgando un servicio de acuerdo a la contribución que hacen al sistema y utilizan Teoría de juegos para obtener soluciones de equilibrio para los jugadores, las cuales no necesariamente darán como resultado una solución óptima para el sistema.

Sin embargo, la Teoría de juegos tiene algunas limitantes que dificultan su aplicación en los juegos, como son [25]:

- Cuando un juego alcanza el equilibrio, éste no necesariamente es un resultado óptimo para el conjunto de jugadores. En el caso del dilema del prisionero, el único equilibrio que se alcanza es (confesar, confesar) lo que pone por más tiempo en prisión a los presos que si decidieran (no confesar, no confesar).
- La teoría de juegos establece que los jugadores se comportan siempre de manera racional. Sin embargo, resultados experimentales en el ámbito de la Economía indican que difícilmente los jugadores en un juego se comportarán completamente racionales [25].
- Una vez alcanzado el equilibrio, los jugadores pueden desear cambiar su estrategia y de hacerlo, el equilibrio se pierde.
- Se puede presentar el caso de múltiples equilibrios en un juego, entonces ¿cómo saber cuál es el equilibrio óptimo para el juego?
- En los juegos repetidos, para determinar que se ha encontrado el equilibrio de Nash en el juego completo, es necesario encontrarlo en cada uno de los subjuegos jugados.

Los puntos antes mencionados han llevado a los investigadores a utilizar herramientas para aplicarlas a la Teoría de juegos, como: computación evolutiva. La computación evolutiva está inspirada en la teoría de la evolución de Darwin [35] y consiste en ejecutar algoritmos

evolutivos para llevar a cabo la optimización de problemas. En el siguiente capítulo describiremos cómo es posible aplicar la computación evolutiva a la Teoría de juegos, específicamente a las estrategias que los jugadores utilizan en los juegos.

Capítulo 4
La evolución de las estrategias en los juegos iterados

En el capítulo anterior describimos los trabajos de investigación encontrados, en los cuales se proponen juegos que se ejecutan de manera iterada y utilizan Teoría de juegos para analizar la situación y predecir cómo se resuelven. Los resultados que se obtuvieron son equilibrios de Nash, lo cual se refiere a encontrar las estrategias de equilibrio para los jugadores en el juego. Las estrategias encontradas, asumen que los individuos se comportan de manera racional en cada iteración del juego, sin embargo, en la realidad las personas involucradas en una situación estratégica más bien toman en cuenta la experiencia obtenida de enfrentarse a la misma situación de manera iterada [25].

Una alternativa que permite el estudio de problemas en donde los involucrados se encuentran ante una situación de manera iterada es la computación evolutiva, la cual se basa en los procesos de la biología evolutiva, enunciados en la teoría de la evolución de Darwin [35].

La computación evolutiva consiste en ejecutar algoritmos evolutivos para la optimización de problemas. Estos algoritmos se basan en el concepto de población, la cual está constituida de individuos o cromosomas, donde cada individuo es una posible solución al problema que se está estudiando [41]. Estos conjuntos consisten de un gran número de soluciones, lo que implica una búsqueda exhaustiva en un gran espacio de soluciones para encontrar la solución óptima al problema.

Con la aplicación de un algoritmo evolutivo a un problema que tiene un conjunto amplio de soluciones, no se asegura encontrar la solución óptima, sin embargo, se asegura encontrar un subconjunto de soluciones que se aproximen a la óptima en un tiempo razonable. A continuación describiremos con detalle en qué consisten los algoritmos evolutivos.

4.1. Algoritmos evolutivos

Un algoritmo evolutivo tiene el siguiente flujo de ejecución, el cual se muestra también en la figura 4.1 [41]:

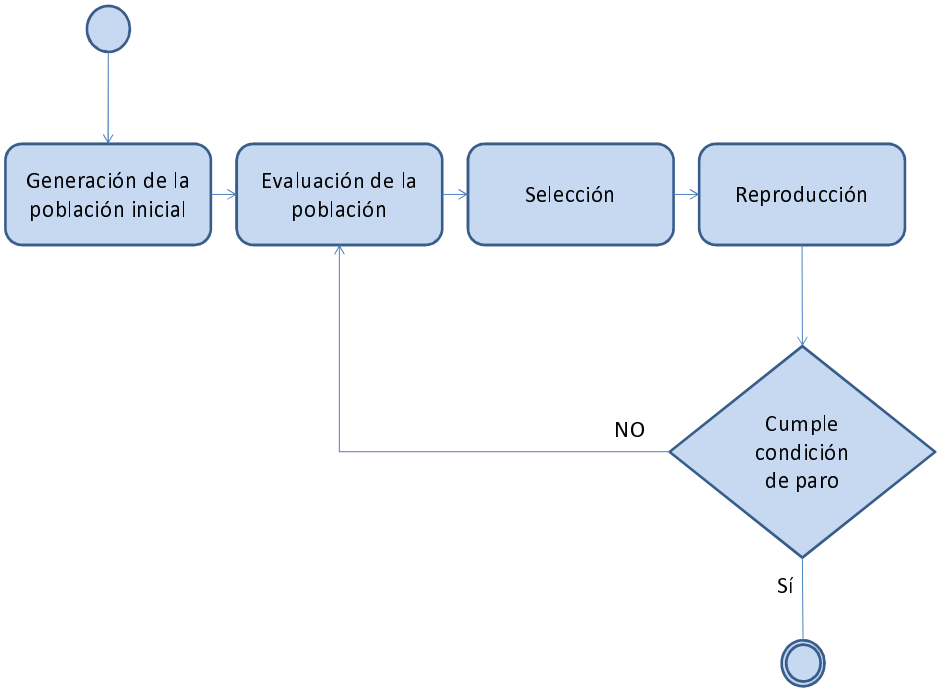

Figura 4.1: Diagrama de flujo de un algoritmo evolutivo

1. Generación de la población inicial.
2. Evaluación de la población.
3. Selección de individuos.
4. Reproducción.

4.1.1. Generación de población inicial

Esta acción se lleva a cabo sólo una vez al inicio de la ejecución del algoritmo y se refiere a encontrar un conjunto de soluciones al problema, la cual recibe el nombre de población, que permita iniciar la ejecución de los pasos 2-4, los cuales se repetirán un número determinado de veces. También en este punto es necesario encontrar la adecuada codificación de los individuos de la población para el problema que se está estudiando. A cada iteración del algoritmo se le da el nombre de generación.

4.1.2. Evaluación de la población

La evaluación de la población se lleva a cabo calculando la aptitud de cada individuo, por medio de una función afin al problema que se está estudiando, la cual permitirá identificar a los mejores individuos de la población.

4.1.3. Selección de los individuos

Una vez evaluados los individuos, se lleva a cabo la selección de aquellos que generarán la población para la siguiente generación. En la selección se utiliza alguna técnica, como [35] [39]: ruleta, torneo, escalamiento sigma, entre otros. A continuación daremos una breve descripción de los tres mecanismos de selección antes mencionados, ya que se utilizarán posteriormente en este trabajo de investigación.

Ruleta

La selección por el método de la ruleta fue introducido por Keneeth Alan De Jong [40] en su tesis de doctorado y es uno de los primeros métodos de selección utilizados en algoritmos genéticos. Este método de selección consiste en los siguientes pasos:

1 Calcular la suma de las aptitudes, S_{tot}, de todos los individuos de la población.

2 Una vez calculada la suma total de la aptitud de los individuos, se elige de manera aleatoria un número $x \in [0, S_{tot}]$.

3 Se recorre el conjunto de individos calculando para cada uno de ellos la suma acumulada de la aptitud, es decir, el primer individuo I_1 tendrá la suma acumulada $S_{a_1} = A_1$, donde A_1 es la aptitud del individuo I_1 y el n-ésimo individuo I_n tendrá la suma acumulada $S_{a_n} = \sum_{i=1}^{n} A_i$, donde A_i es la aptitud del individuo I_i. El primer individuo cuya suma acumulada exceda el valor de x, es seleccionado.

Torneo

Este método de selección se basa en la elección aleatoria de un grupo de individuos, generalmente dos. Una vez seleccionados los individuos, se compara la aptitud que tiene cada uno de ellos y se elige al individuo con mayor aptitud en el grupo para formar una pareja que producirá nuevos individuos para la nueva población. Este procedimiento se repite hasta completar las parejas que se necesitan para generar la población de la siguiente generación.

Escalamiento Sigma

La selección por escalamiento sigma permite mapear la aptitud de los individos con su valor esperado el cual indica el número de descendientes que puede producir cada uno de ellos, de tal forma que no se presente convergencia prematura. El valor esperado de cada

individuo estará en función de su aptitud y de la desviación estándar de la población. La ecuación para calcular el valor esperado de los individuos es la siguiente [39]:

$$V_{e_i}(i,t) = \begin{cases} \frac{f(i)-\bar{f(t)}}{\sigma(t)} & si \quad \sigma(t) \neq 0 \\ 0 & en\ otro\ caso \end{cases} \qquad (4.1)$$

donde:

- $f(i)$: aptitud de individuo i.
- $\bar{f(t)}$: aptitud promedio de la población en la generación t;
- $\sigma(t)$: la desviación estandar de la aptitud de la población en la generación t.

4.1.4. Reproducción

Un algoritmo evolutivo utiliza la reproducción de los individuos de la generación actual, para generar una nueva población para la siguiente generación. Esto se lleva a cabo utilizando diferentes métodos, como [41]: duplicación, mutación y cruza.

Duplicación

En la duplicación, se generan hijos que son copias idénticas de uno de lo padres, como por ejemplo la división celular, en donde las células nuevas son idénticas a la que inicia el proceso de duplicación.

Mutación

La mutación en la población se lleva a cabo sobre un determinado número de individuos para convertirlos en mutantes y observar el impacto que producen en la población. La mutación es el proceso de cambiar uno o más elementos (gen) de un cromosoma. La figura 4.2 muestra la mutación en un cromosoma.

Cruza

Para llevar a cabo la cruza, se seleccionan dos individuos con alguna de las técnicas mencionadas en la sección 4.1.3. Una vez obtenidas las parejas necesarias para producir la población de la siguiente generación, se lleva a cabo la cruza. Existen varias formas de cruza: en un punto, dos puntos y uniforme.

- Cruza en un punto: se elige de manera aleatoria un punto para la cruza, cada individuo de la pareja se divide en el punto elegido. La primera parte del primer individuo formará parte del cromosoma de primer hijo y la segunda parte del segundo individuo completará el cromosoma del primer descendiente. La primera parte del segundo individuo constituye la primera parte del cromosoma del segundo hijo y la segunda parte del primer individuo completará el segundo cromosoma (Ver figura 4.3.a).

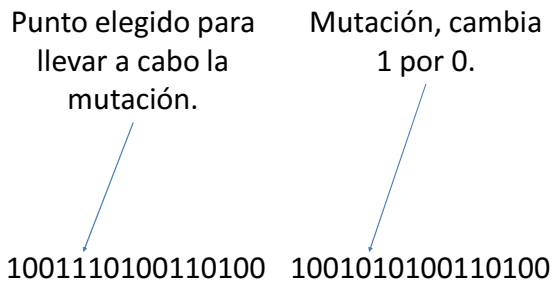

Figura 4.2: Mutación de un cromosoma

- Cruza en dos puntos: en la cruza de dos puntos, se eligen de manera aleatoria dos posiciones para llevar a cabo la cruza, una vez elegidas las posiciones, se lleva a cabo la recombinación de las partes en que se dividen los individuos padre para generar a los hijos. La figura 4.3.b muestra cómo se lleva a cabo la cruza. Es importante mencionar que la cruza puede generalizarse a n-puntos.

- Cruza uniforme: se genera de manera aleatoria una cadena binaria y cada elemento de esta cadena indicará de qué padre será tomado el siguiente gen para el primer hijo. Si el valor de la cadena es 1 entonces se elige un gen del primer padre y si el valor es 0 entonces se elegirá un gen del segundo padre. Para construir el cromosoma del segundo hijo, se invierten los valores de la cadena binaria y se realiza el mismo procedimiento. La figura 4.3.c muestra cómo se lleva a cabo la cruza uniforme.

Es importante mencionar que la condición de paro para el algoritmo genético, dependerá del problema que se esté estudiando y puede consistir en encontrar una solución del problema cercana a la óptima o iterar el algoritmo un número determinado de generaciones y observar el resultado obtenido.

Dentro de los algoritmos evolutivos se tiene la siguiente clasificación:

- Algoritmos Genéticos: son algoritmos que se basan en la selección natural y la genética. La codificación de los individuos o cromosomas se lleva a cabo utilizando cadenas binarias y el principal operador de reproducción es la cruza. Se asume que todos los individuos generados son de la misma especie.

- Estrategias evolutivas: son algoritmos que ponen énfasis en la relación que hay entre el comportamiento que tiene un individuo y su descendencia. La codificación de los individuos utiliza números reales y el principal operador de reproduccón es la mutación.

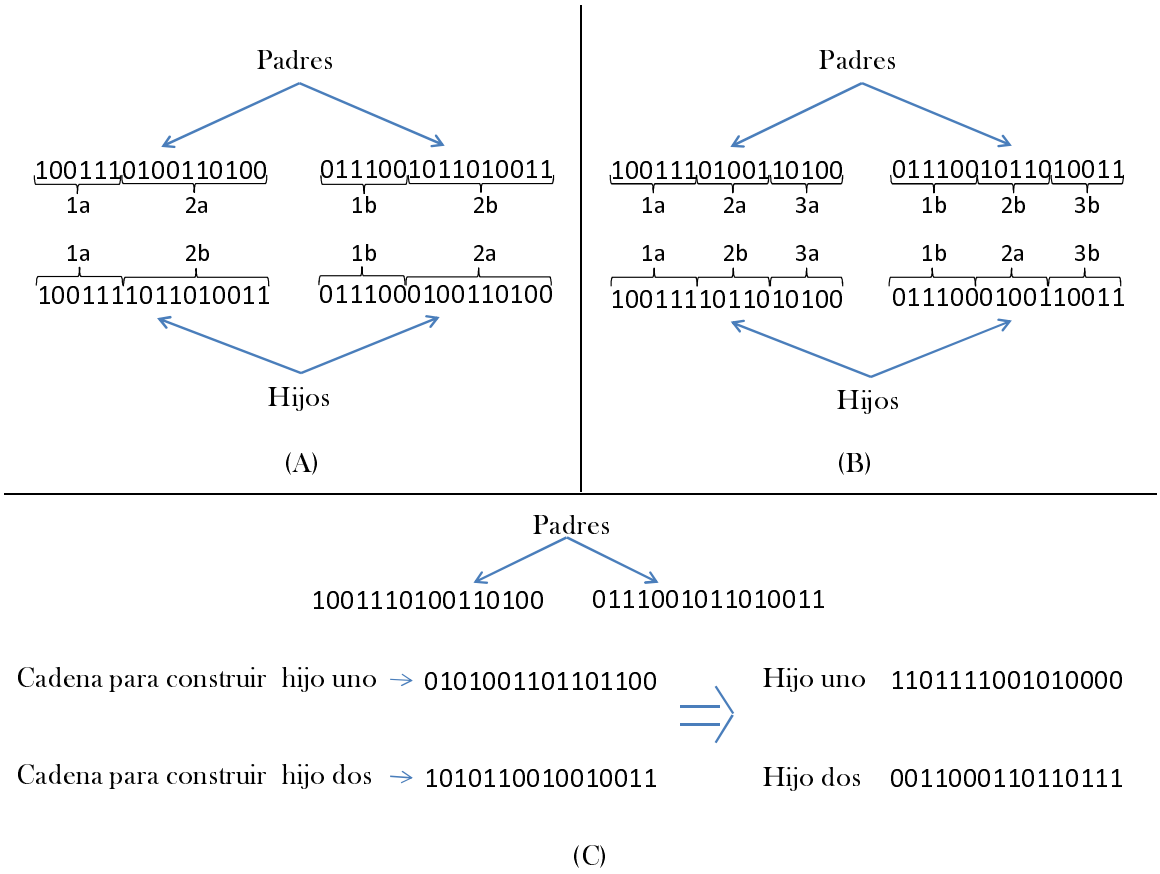

Figura 4.3: Diferentes técnicas de cruza: (A) un punto, (B)dos puntos, (C) uniforme.

Al igual que en los algoritmos genéticos, los individuos de la población pertenecen a una misma especie.

- Programación evolutiva: la representación de los individuos se lleva a cabo por medio de un autómata finito, donde cada estado representa a un individuo de la población, y éste a su vez una especie, es decir, la población estará constituida por tantas especies como individuos tenga. Su principal operador de reproducción es la mutación.

Es importante recalcar que los tres tipos de algoritmos evolutivos antes mencionados, no son los únicos, pero son los más reconocidos [41]. En la siguiente sección describiremos con detalle en qué consiste un algoritmo genético, ya que será de utilidad para el resto de este capítulo, así como para llevar a cabo el desarrollo de este trabajo de investigación.

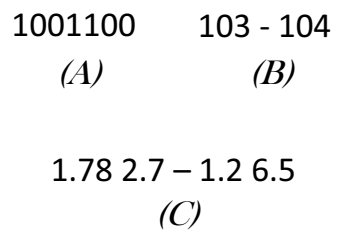

Figura 4.4: Representación de un cromosoma: (a)binaria, (b)entera, (c)decimal.

No. Individuo	Población Inicial	Valor Entero (x)
1	01101	13
2	11000	24
3	01000	8
4	10011	19

Tabla 4.1: Representación de posibles soluciones al problema, como cadenas binarias (cromosomas)

4.2. Algoritmo genético

Los algoritmos genéticos tienen sus inicios en los años 50's con el trabajo realizado por el científico John Henry Holland y son comúnmente utilizados para resolver problemas de optimización combinatoria, los cuales tienen un amplio conjunto de soluciones, también son utilizados en las áreas de aprendizaje maquinal, ecología, genética de poblaciones y sistemas sociales, entre otros.

Un algoritmo genético consiste en codificar a los individuos de una población como cadenas de bits, las cuales pueden contener elementos binarios, enteros o decimales (Ver fig. 4.4).

A continuación describiremos el funcionamiento de un algoritmo genético, utilizando un ejemplo: encontrar el máximo de la función $f(x) = x^2$.

4.2.1. Aplicación de algoritmos genéticos: *Encontrar el máximo de la función* $f(x) = x^2$

Generación de la población inicial

Para encontrar el máximo de la función $f(x) = x^2$ sobre los enteros (1,2,3,...,32), la tabla 4.1 muestra la representación de un conjunto de soluciones como cadenas binarias, las cuales son generadas de manera aleatoria.

Población Inicial	Valor Entero (x)	$f(x) = x^2$
01101	13	169
11000	24	576
01000	8	64
10011	19	361

Tabla 4.2: Evaluación de la población cuya función de aptitud es $f(x) = x^2$

Población Inicial	Valor Entero (x)	$f(x) = x^2$	ValorE	Parejas
01101	13	169	0.726144557	1
11000	24	576	1.628647921	1
01000	8	64	0.493311993	1
10011	19	361	1.151895529	1
Promedio = 292.5				
Desviación estandar (σ) : 225,483924				

Tabla 4.3: Muestra el valor esperado que obtiene cada individuo de la población

Evaluación de la población

Una vez que se tiene la población inicial, esta es evaluada. La función que nos permite conocer cuál es la aptitud de cada individuo de la tabla 4.1, es $f(x) = x^2$. La tabla 4.2 muestra los resultados que obtiene cada individuo y podemos ver que el individuo cuyo cromosoma es 11000 es el que tiene mayor aptitud.

Selección de individuos

Para este ejemplo utilizamos el método de selección por escalamiento sigma. Para ello calculamos el promedio de la aptitud de los individuos, así como la desviación estándar. Con los valores obtenidos se obtiene el valor esperado de cada individuo, el cual nos indica el número de descendientes que pueden tener. La tabla 4.3 muestra el valor esperado de los individuos.

Una vez que se tiene el valor esperado de cada individuo, se obtiene el número de parejas para cruza que tendrá cada individuo. Aquellos individuos cuyo valor esperado está entre $[-\sigma, \sigma]$ tendrán sólo una pareja. Si el valor esperado está por arriba de σ, entonces recibirán dos parejas. Los resultados se muestran en la tabla 4.3.

Reproducción

En este ejemplo utilizaremos la cruza en un punto y la mutación para todos los individuos. Como primer paso generamos las parejas que producirán la población de la siguiente generación de manera aleatoria (Ver tabla 4.4) .

Pareja	Indivudo 1	Individuo 2	Punto Cruza
1	01101	11000	2
2	10011	11000	3

Tabla 4.4: Parejas seleccionadas para generar los individuos de la siguiente generación.

Individuo	Cromosoma	Valor Entero	$f(x) = x^2$
1	01000	8	64
2	11101	29	841
3	10000	16	256
4	11011	27	729

Tabla 4.5: Población que se considerará para la siguiente generación.

Una vez que se tienen las parejas, se elige un punto para llevar a cabo la cruza, el cual es diferente para cada pareja. Los descendientes creados se someten al proceso de mutación invirtiendo un elemento de su cromosoma. La nueva población, es decir la siguiente generación, se muestra en la tabla 4.5.

El procedimiento antes descrito, se repite un número determinado de veces de acuerdo al criterio de paro que se haya elegido.

Una investigación sobre sistemas sociales que aplica algoritmos genéticos fue la que realizó el Dr. Robert Axelrod, y que lleva por nombre "Evolving New Strategies: The Evolution of Strategies in the Iterated Prisoner's Dilemma"[37]. En la siguiente sección describiremos cómo se llevó a cabo este trabajo y los resultados obtenidos.

4.3. El dilema del prisionero iterado de Axelrod

El Dr. Robert Axelrod autor del libro: *"La evolución de la cooperación"* [38], hizo un estudio del juego del dilema del prisionero de manera repetida, es decir, simuló el hecho de que los jugadores del Dilema del prisionero tienen que enfrentarse a la decisón de cooperar o traicionar repetidamente. Llevó a cabo esta simulación utilizando algoritmos genéticos, pues asegura que los jugadores o individuos que se encuentran en una situación estratégica, no son capaces en su totalidad de analizar y calcular sus estrategias óptimas, más bien cada individuo actualiza sus estrategias basándose en resultados anteriores.

En primer lugar estableció cómo representar cada estrategia posible (cooperar, traicionar) en un juego de dos jugadores para una sola tirada. La estrategia de cooperar es representada por una C y la estrategia de traicionar (no cooperar) con una D, teniendo entonces cuatro

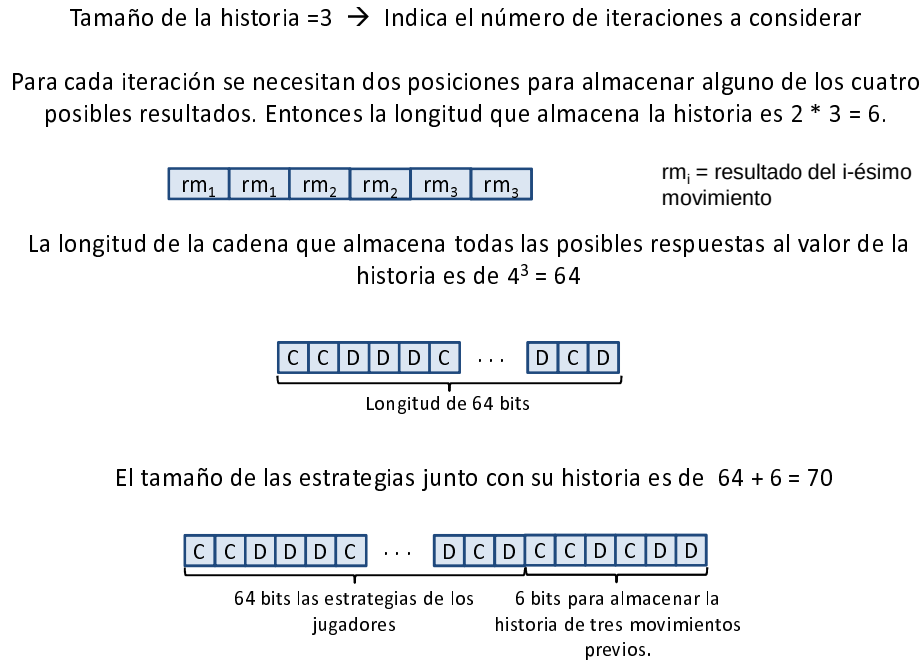

Figura 4.5: Cromosomas para el dilema del prisionero iterado.

posibles resultados: CC, CD, DC y DD.

Axelrod consideró que los jugadores eligen su decisión C o D en función de los movimientos previos realizados en juegos anteriores, a lo que le dió el nombre de historia. La historia puede contener un número de movimientos variable. Por ejemplo si se considera la historia de un movimiento previo, entonces se tienen cuatro historias diferentes: CC, CD, DC y DD y como resultado un conjunto de $2^{4^1} = 16$ estrategias diferentes.

Para las simulaciones del dilema del prisionero iterado realizadas en [37], se considera que los individuos tomarán su siguiente decisión en función de la historia de tres movimientos previos, teniendo entonces $4 * 4 * 4 = 64$ historias diferentes de 3 movimientos anteriores, por lo tanto se tienen $2^{4^3} = 2^{64}$ estrategias diferentes. Considerando que para el inicio del juego se debe de contar con una historia ficticia de tres movimientos, entonces el número de estrategias aumenta a $2^{64+6} = 2^{70}$. Las estrategias obtenidas representan a los individuos en el juego (ver figura 4.5).

Una vez establecidas las estrategias en el juego, el algoritmo genético se ejecuta de la siguiente manera:

- **Generación de la población inicial:** se elige aleatoriamente un conjunto inicial de 20 individuos (estrategias), que son cadenas de 70 C's y D's. Se elige un conjunto inicial aleatorio debido a que buscar las mejores estrategias de entre 2^{70} es una tarea complicada.
- **Evaluación de la población:** en cada iteración cada individuo calcula su efectividad, que es el puntaje promedio que obtiene un jugador de todos los juegos que haya jugado.
- **Selección de los individuos:** los individuos más aptos son seleccionados para tener descendientes, y para ello se utilizó la técnica de escalamiento sigma la cual fué descrita en la sección 5.1.3. Aquellos individuos que se encuentran dentro del promedio se cruzan una vez, los que están por arriba de una desviación estándar se cruzan dos veces y aquellos que están por debajo de una desviación estándar no obtienen cruza alguna.
- **Reproducción:** la reproducción de los individuos se lleva a cabo utilizando la cruza en un punto. Aquellos individuos que obtuvieron dos cruzas producen cuatro descendientes y los que obtuvieron una cruza producen dos. También se aplica mutación a los individuos de la población de manera aleatoria, cambiando una pequeña proporción de C's por D's y viceversa.

Al final de una ejecución, la nueva generación tendrá patrones de comportamiento heredados de los individuos más exitosos de la población anterior. El algoritmo se encuentra descrito con más detalle en [37].

Los resultados que se obtuvieron, reflejan que las estrategias que evolucionaron son semejantes a la que se conoce como "Tit for Tat", en la cual los jugadores en un inicio cooperan y en las subsecuentes confrontaciones se comportarán de acuerdo a cómo se comportó su oponente en el juego anterior. Los comportamientos que se observaron fueron los siguientes:

- Seguir cooperando después de tres movimientos de cooperación mutua.
- Traicionar cuando el oponente traiciona sin razón alguna.
- Seguir cooperando después de restaurada la cooperación.
- Cooperar una vez que se restaura la cooperación aún y cuando ha sido explotado.
- Traicionar después de tres movimientos de traición mutua.

4.4. Discusión

En este capítulo describimos el funcionamiento de los algoritmos genéticos y observamos el trabajo que realizó el Dr. Robert Axelrod, en el cual estudia la situación estratégica de los jugadores en el juego del Dilema del Prisionero, cuando ésta se efectúa de manera repetida.

Si observamos que los nodos de un sistema distribuido P2P se encuentra de manera repetida en la situación de decidir cooperar o no con el sistema al que pertenecen, y que sus decisiones cambiarán en función de la experiencia que han obtenido, repercutiendo en el beneficio propio y en el del sistema, podemos entonces hacer uso de los algoritmos genéticos para estudiar cómo cambia el comportamiento de los nodos en el sistema.

En el siguiente capítulo describiremos una propuesta que nos permite estudiar el comportamiento de los nodos de un sistema, cuando se ven involucrados en una situación de toma de decisiones, la cuál se propone como un juego describiendo los componentes de mismo, como se lleva a cabo en la teoría de juegos y aplicamos algoritmos genéticos para observar la evolución de la cooperación en la población que compone el sistema.

Capítulo 5
Evolución de la cooperación para la duplicación utilizando algoritmos genéticos

En este capítulo proponemos un juego que nos permite modelar la situación estratégica de los nodos, al encontrarse bajo el dilema de cooperar o no con el sistema al que pertenecen. Proponemos también, dos mecanismos de compensación que tienen como objetivo gratificar a los nodos de un sistema P2P, por cooperar duplicando documentos y así, alcanzar la disponibilidad que requieren. Aplicamos, al igual que en el trabajo realizado por Robert Axelrod [37] el cual fue descrito en el capítulo anterior, algoritmos genéticos para observar la evolución del comportamiento de los nodos que se encuentran bajo la influencia de los mecanismos de compensación.

5.1. Problema

Los sistemas P2P tienen una característica fundamental: están constituidos por nodos autónomos que deciden qué, cuándo y cuánto cooperar con recursos en el sistema. Diferentes estudios realizados en algunos sistemas P2P, como Gnutella, Maze P2P y eDonkey [26], indican que la mayoría de los nodos solo consumen recursos del sistema y no comparten los propios y además que sólo un pequeño porcentaje distribuye la mayoría de los recursos solicitados. Esta situación pone en riesgo los recursos contenidos en un sistema, ya que si sólo unos cuantos ponen a disposición los recursos que almacenan de manera desinteresada, mientras la gran mayoría sólo consume, entonces se pone en riesgo la disponibilidad de los mismos.

Debido a la situación antes descrita, se han realizado varios trabajos de investigación en los cuales se proponen mecanismos que permitan incrementar la cooperación de los nodos en los sistemas P2P. Algunos de estos trabajos, como los mencionados en la sección 3.5 utilizaron Teoría de juegos para incentivar a los nodos a cooperar con el sistema. Si bien los resultados que se obtuvieron eran los deseados, también es cierto que se suponía que los nodos siempre tenían un comportamiento racional, así como en algunos casos, conocimiento total de las decisiones tomadas por el resto de los nodos.

Sin embargo, estudios científicos en Economía indican que los involucrados en una situación de toma de decisiones no se comportan completamente racionales al momento de tomarlas [25], además de que en un sistema P2P, cuyas conexiones entre nodos es variable, difícilmente se cumple el hecho de que cada nodo tiene conocimiento de lo que hace cada uno de los otros nodos en el sistema [42].

De lo anterior, surge la siguiente pregunta: ¿cómo motivar, que los nodos de un sistema P2P cooperen con recursos, considerando que su comportamiento influye tanto en el beneficio propio como de manera global en el sistema?

Esta interrogante determina el objetivo de este trabajo de investigación, así como las características de los mecanismos que proponemos.

5.2. Objetivo general

Observar cómo se ve afectada la evolución del comportamiento de los nodos de un sistema P2P, cuando se encuentran bajo esquemas o mecanismos de compensación que buscan motivar su cooperación (duplicación de documentos), y cómo este comportamiento afecta el beneficio global del sistema.

5.3. Metodología

1. Estudiar las características de los sistemas P2P.
2. Estudiar la Teoría de juegos, así como de los trabajos de investigación realizados en sistemas P2P en los cuales se aplicó esta teoría.
3. Diseñar mecanismos de compensación que evalúen la cooperación de los nodos con respecto a su contribución de recursos (duplicación de documentos) en un sistema P2P.
4. Diseñar un juego que permita modelar la situación estrategia (duplicar o no duplicar) de los nodos y el sistema P2P.
5. Estudio de algoritmos genéticos.
6. Implementación del juego propuesto, aplicando los mecanismos propuestos y algoritmos genéticos.
7. Experimentación y análisis de los resultados obtenidos.

El primer punto fue descrito en el capítulo 2, los puntos 2 y 3 en los capítulos 3 y 4. Los puntos 4-7, se presentarán en los capítulos 5-7.

Umbral de Pago	Costo por Consulta
x	$\frac{x}{2}$
x	x
x	$2x$

Tabla 5.1: Distintos valores de costo por consulta de documentos, en relación con el umbral de pago de los nodos.

5.4. Diseño de mecanismos de compensación para motivar la cooperación de la duplicación

Proponemos dos mecanismos de compensación, con los cuales los nodos que cooperan con el sistema se ven beneficiados por su comportamiento. El beneficio que obtienen influye directamente en la cantidad de recursos (documentos) que pueden consumir en el sistema. A continuación describimos los mecanismos propuestos.

5.4.1. Mecanismo de compensación económica

Este mecanismo consiste en compensar económicamente a los nodos que almacenan un documento, donde dicha compensación será otorgada por el solicitante de la duplicación. Cada nodo contará con un umbral mínimo de pago, el cual puede cambiar la decisión de cooperar de un nodo, si la compensación económica está por debajo de su umbral.

Si los nodos desean consultar algún documento en el sistema, tendrán que pagar un costo para llevar a cabo la consulta, de tal forma que aquellos que dupliquen una cantidad alta de documentos, tendrán mayor oportunidad de consultar los documentos que deseen. El costo que debe pagar un nodo por consultar un documento, está en función de su umbral de pago mínimo. La tabla 5.1 muestra tres distintas maneras de relacionar el umbral de pago mínimo que tiene cada nodo, con el costo que le implica consultar un documento.

5.4.2. Mecanismo de compensación por reconocimiento social

Este mecanismo consiste en dar una calificación a los nodos de acuerdo al comportamiento que muestran a los documentos que buscan espacio de almacenamiento en el sistema. Esta calificación se otorga en función del número de veces que cooperaron con el sistema ($\#coop$) y el número total de documentos que solicitaron espacio de almacenamiento ($\#Docts_{tot}$). La siguiente ecuación muestra cómo se calcula la calificación de los nodos.

$$Cal_i = (\frac{\#coop}{\#Docts_{tot}}) * 10 \tag{5.1}$$

Calificación	% Consultas
$calificación > 8$	%100
$8 \geq calificación > 6$	%80
$6 \geq calificación > 4$	%60
$4 \geq calificación > 2$	%40
$2 \geq calificación > 0$	%20
$calificación = 0$	%0

Tabla 5.2: Distintos porcentajes de consulta de documentos, en función de la calificación obtenida para cada nodo.

Los nodos que deseen consultar documentos almacenados en el sistema, tendrán acceso sólo a un porcentaje de los documentos que quieren consultar, el cual estará determinado por la calificación que obtuvieron. Entre más cercana a 10 sea la calificación, el nodo podrá consultar un mayor número de documentos. La tabla 5.2 muestra la cantidad de documentos que podrá consultar un nodo por la calificación obtenida.

En la siguiente sección describiremos un juego en el cual se refleja la situación estratégica de los nodos, la cual consiste en duplicar o no los documentos que solicitan espacio de almacenamiento al sistema.

5.5. Diseño del juego de duplicación

Proponemos un juego en el cual se simula la interacción entre los nodos de un sistema P2P y un conjunto de documentos que buscan espacio de almacenamiento para ser duplicados. A continuación describiremos el juego propuesto.

5.5.1. Descripción del juego

Con el juego que a continuación se describe, se busca que el sistema pueda almacenar el mayor número de documentos posible para que, tanto el sistema, como los nodos que lo conforman se vean beneficiados de este almacenamiento, por la aplicación de los mecanismos de compensación descritos en la sección anterior.

Componentes del juego

- Jugadores: nodos.
- Estrategias: duplicar y no duplicar.
- Utilidad o Beneficio: se mide en función del número de documentos que los nodos pueden consultar.

La siguiente ecuación nos permite calcular la razón de documentos que un nodo pudo consultar.

$$U_i = (\frac{\#CE_i}{\#CT_i}) \tag{5.2}$$

El $(\#CE_i)$ se refiere al número de consultas exitosas que pudo llevar a cabo un nodo y $(\#CT_i)$ el número total de consultas que quiso realizar. El algoritmo que se muestra en la figura 5.1, muestra cómo se lleva a cabo la ejecución del juego de duplicación.

Para cada documento que envía una solicitud de espacio de almacenamiento al sistema, se realiza lo siguiente:

1. Cada nodo en el sistema decide duplicar o no el documento y sólo responden aquellos que están interesados.

2. El sistema calcula la disponibilidad que obtiene el documento con los nodos que decidieron almacenar. Si la disponibilidad obtenida es mayor o igual a la que requiere el documento, entonces se queda en el sistema. De lo contrario el sistema pierde la posibilidad de almacenar el documento.

3. Si el documento se queda en el sistema, se llevará a cabo la elección de los nodos necesarios para cubrir la disponibilidad que se requiere. Estos nodos serán elegidos del conjunto de nodos que decidió almacenar el documento.

4. Una vez elegidos los nodos que almacenarán el documento, entonces recibirán una compensación por su cooperación al sistema.

Los pasos 1-4 se repiten un número determinado de veces, con diferentes documentos. Una vez que un conjunto de documentos ha pasado por el sistema solicitando espacio de almacenamiento, en el caso de la compensación económica todos los nodos observan la cantidad de dinero acumulado y en el caso del reconocimiento social, para cada nodo en el sistema se calcula la calificación en función de su cooperación.

Los nodos interesados en realizar consultas a los documentos que se encuentran en el sistema, lo harán en función del capital acumulado cuando se utiliza el mecanismo de compensación económica y en la calificación obtenida en el mecanismo de reconocimiento social.

Una vez que los nodos han efectuado las consultas que su capital y calificación les han permitido, los nodos calculan el beneficio o utilidad obtenida utilizando la ecuación antes descrita.

Para calcular la disponibilidad que se proveerá a los documentos, utilizamos la ecuación descrita en [13]:

$$A_j = 1 - (1 - a_j)^{S_j}, \quad 0 \leq j \leq M \tag{5.3}$$

donde

```
procedimiento Juego()
    listaDocumentos;
    listaDocumentosAlmacenados;
    listaNodos;

    Para i = 1 hasta m hacer
        //Se obtiene el siguiente documento de la lista
        documento ← obtieneDocumento(listaDocumentos ,i);
        Para j = 1 hasta n hacer
            //Se obtiene el primer nodo
            nodo ← obtieneNodo(listaNodos , j);
            tomaDecision(nodo, documento);
        FinPara

        //Se calcula la disponibilidad alcanzada con los nodos cuya decisión es duplicar
        alcanzaDisp = calculaDisp(listaNodos);
        Si (alcanzaDisp == true) entonces
            //Se lleva a cabo la elección de los nodos entre aquellos cuya decisión es duplicar
            eleccion(listaNodosCoop, tipoEleccion);
            //Actualiza la utilidad y la variable de cooperaciones, así como la historia
            actualizaDatosNodos(listaNodos);
        FinSi
    FinPara

    actualizaCalificacion(listaNodos);
    consultaDocumentos(listaNodo, listaDocumentosAlmacenados);
FinProcedimiento
```

Figura 5.1: Algoritmo de ejecución para el juego de duplicación.

- j - Se refiere a j-ésimo documento que busca espacio de almacenamiento.
- a_j - Indica la disponibilidad promedio de los nodos que con seguridad almacenarán el documento j.
- S_j - Indica el número de nodos que almacenarán el objeto j.

Es importante mencionar que en la ecuación 5.3, en [13] el valor de a_j es el promedio de la disponibilidad de todos los nodos en el sistema. Para este trabajo de investigación a_j es el promedio de la disponibilidad de los nodos que almacenarán el documento.

A continuación describiremos cómo se llevará a cabo la elección de los nodos.

5.5.2. Elección de los nodos

Para elegir a los nodos que almacenarán el documento, de aquellos que mostraron interés, se proponen tres técnicas: secuencial, aleatoria y round-robin.

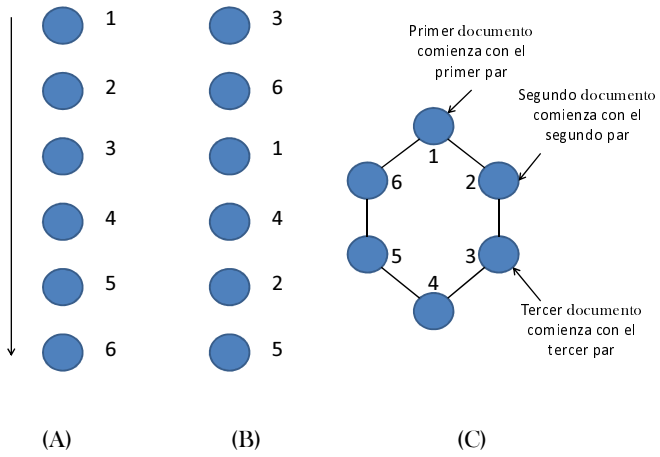

Figura 5.2: Técnicas de selección: (A) Secuencial, (B) Aleatoria, (C) Turno Rotatorio

Elección secuencial

Esta se lleva a cabo colocando a los nodos interesados en una lista y, siempre que un documento solicita espacio de almacenamiento, la elección comienza por el primer elemento de la lista, deteniéndose una vez que se haya completado la cantidad de nodos necesarios para proporcionar la disponibilidad que el documento requiere. En la figura 5.2.(A) se observa cómo se lleva a cabo esta elección.

Elección aleatoria

La figura 5.2.(B) muestra el tipo de elección aleatoria, en la cual una vez que se tiene el conjunto de nodos que quieren almacenar el documento, se genera un número de manera aleatoria, tal que $x \in [1, tamLst]$ y se elige el nodo cuya posición en la lista corresponde a x. Esto se lleva a cabo de manera repetida hasta completar los nodos que proporcionen la disponibilidad requerida por el documento.

Elección turno rotatorio

La elección por turno rotatorio, consiste en considerar el número de documentos que solicitan espacio de almacenamiento. Cuando llega un primer documento solicitante, la elección de los nodos comenzará por el primer nodo que se encuentra en la lista de aquellos que quieren almacenar. A la llegada de un segundo documento, la elección comenzará por el segundo nodo de la lista, y así para cada documento que solicite espacio de almacenamiento, como se muestra en la figura 5.2.(C).

5.6. Aplicación de algoritmos genéticos a los mecanismos de compensación y al juego de duplicación

Como ya describimos en la sección 4.1, para poder aplicar algoritmos genéticos a un determinado problema, es necesario encontrar la codificación que permita representar el problema que se está estudiando. Para esta investigación nos inspiramos en el trabajo realizado por el Dr. Robert Axelrod, el cual fue descrito en la sección 4.2.

Generación de la población inicial: Para la generación de la población estamos considerando que los nodos son representados por cromosomas y que cada uno de estos representa una estrategia en el juego. Los cromosomas son cadenas binarias constituidas de "0" y "1", lo que indica al nodo duplicar o no.

Para almacenar el resultado de una iteración, son necesarios tres bits que contendrán los siguientes elementos:

$$< duplica, queda, elegido >$$

donde:

- *duplica* toma el valor de 1 si el nodo quiere duplicar el documento y 0 en otro caso.
- *queda* toma el valor de 1 si el documento alcanza la disponibilidad requerida y 0 en otro caso.
- *elegido* toma el valor de 1 si el nodo es electo para duplicar y 0 en otro caso.

es importante reiterar que si el documento no se queda en el sistema, entonces la elección de los nodos no se llevará a cabo.

Tenemos entonces que para una interacción se requieren $3 * 1$ bits para almacenar el resultado. Si al número de interacciones lo llamamos k, entonces es necesario $3k$ bits para almacenar la reciente historia h. Esta historia se refrescará cada vez que se lleve a cabo una interacción, es decir cada vez que un documento solicite ser duplicado.

Ahora bien una estrategia debe definir una respuesta para cada posible valor de h, por lo tanto se necesitan 2^{3k} para representar estas respuestas, así cada estrategia es codificada en una cadena binaria de longitud $3k + 2^{3k}$.

Un ejemplo se muestra en la figura 5.3, en donde consideramos una $k = 2$, entonces tendremos un historia $h = 3 * 2 = 6$ y el tamaño de las estrategias será de $2^{3*2} = 64$, teniendo una cadena final de $6 + 64 = 70$ para cada cromosoma.

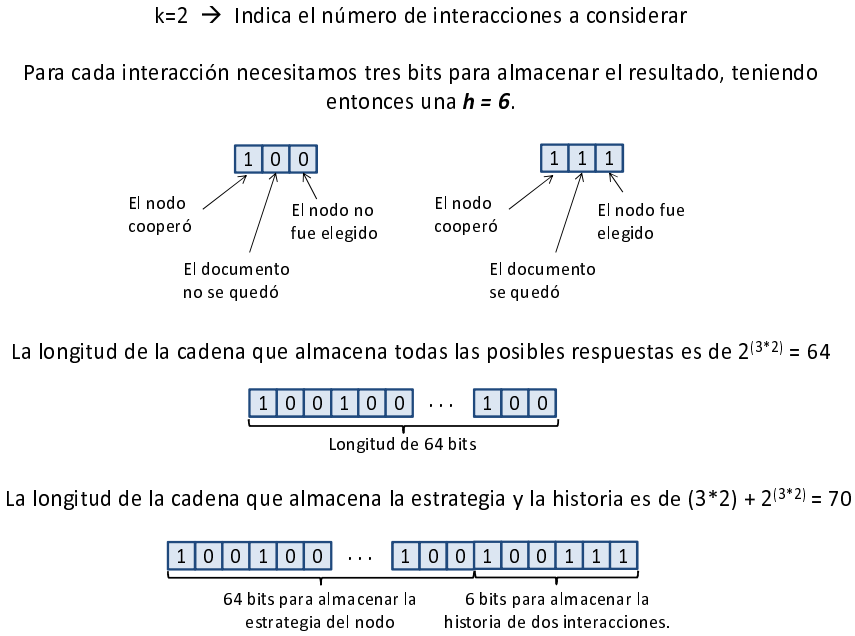

Figura 5.3: Técnica para armar una estrategia (cromosoma).

Evaluación: La evaluación se lleva a cabo, calculando la aptitud de cada nodo en función de los documentos que consultó en el sistema. Para ello utilizamos la ecuación 5.2 de utilidad descrita en la sección 5.5.1, que indica cuál es la aptitud de cada individuo. Una vez evaluados los nodos, también el sistema es evaluado, contabilizando el número de documentos que pudo almacenar al final de la generación.

Selección: La selección de los individuos que formarán la siguiente generación, utiliza el método de selección por escalamiento sigma descrito en la sección 4.1.3.

Reproducción:

Cruza Cuando la cruza se lleva a cabo, los padres, además de heredar a sus descendientes parte de su cromosoma, también heredan dos componentes más del padre con mayor aptitud, las cuales son: disponibilidad y el umbral de pago mínimo para el mecanismo de compensación económica y únicamente el de disponbilidad para el de reconocimiento social. En caso de que la cruza no se realice, entonces se hará una copia de los individuos seleccionados, las cuales formarán parte de la población para la siguiente generación. En la figura 5.4, mostramos cómo se lleva a cabo la cruza de los nodos seleccionados suponiendo una k=1.

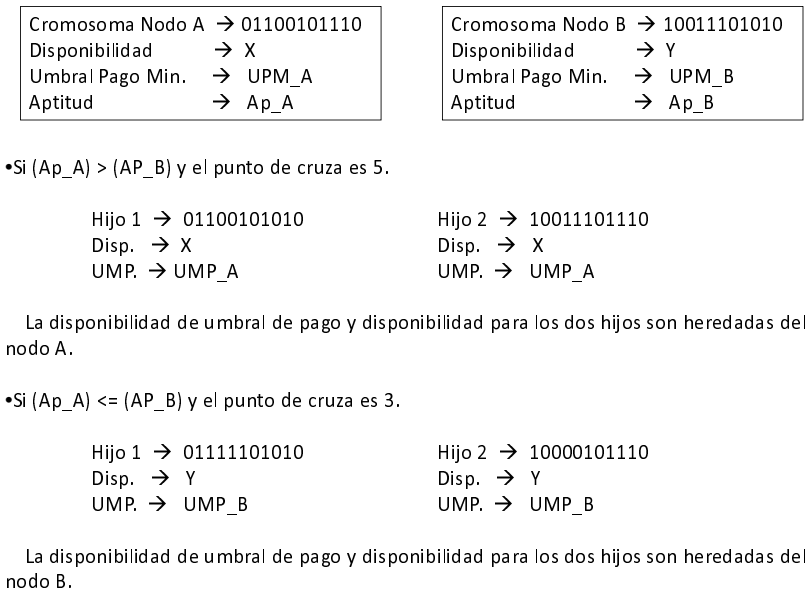

Figura 5.4: Ejemplificación de la cruza para el juego de duplicación.

Mutación La mutación se lleva a cabo de manera uniforme en cada gen del cromosoma, es decir, dada cierta probabilidad, se genera de manera uniforme un número aleatorio, si este es menor que la probabilidad establecida, entonces se muta el gen cambiando de 0 a 1 o viceversa.

Capítulo 6
Experimentación y análisis de resultados

En este capítulo se presenta un conjunto de resultados experimentales que nos permiten observar cómo es afectado el comportamiento de los nodos de un sistema P2P al encontrarse bajo los mecanismos de compensación descritos en el capítulo 5 y analizamos su evolución aplicando algoritmos genéticos. Para llevarlos a cabo, se realizó la implementación del juego de duplicación, los mecanismos de compensación y el algoritmo genético, cuya descripción se encuentra en el apéndice A.

6.1. Descripción general

Dado un conjunto de nodos en un sistema P2P y un conjunto de documentos que buscan espacio de almacenamiento, analizamos el comportamiento de los nodos que se encuentran bajo el mecanismo de (A) reconocimiento social y (B) compensación económica. Cada nodo tomará la decisión de cooperar con espacio de almacenamiento para duplicar un documento, recibiendo una remuneración por su desición. Después de que cierto número de documentos hayan solicitado espacio de almacenamiento, los nodos podrán realizar consultas de lo que se encuentra almacenado en el sistema y en función del éxito que obtiene al consultar, serán evaluados.

El resultado de la evaluación de cada nodo es considerada su aptitud, la cual es utilizada en la ejecución del algoritmo genético.

La finalidad de la experimentación está orientada a:

- Observar cómo evoluciona el comportamiento de los nodos, cuando se encuentran bajo la influencia de los mecanismos de compensación propuestos, considerando como métrica qué mecanismo logra que el sistema almacene un alto porcentaje de los documentos que solicitan ser almacenados.
- Observar, al final de las generaciones, cómo está constituida la población de nodos, es decir, cómo están compuestas sus estrategias y cúal es el comportamiento que muestran.

A continuación describiremos los parámetros que fueron utilizados en la experimentación para la ejecución de las simulaciones.

6.2. Parámetros

6.2.1. Población

Conjunto de nodos que conforman el sistema. A cada nodo de la población se le asigna una estrategia (cromosoma), un valor de disponibilidad y un umbral de pago mínimo (mecanismo de compensación económica).

Tipo de Nodo

Consideramos tres tipos de nodos: altruistas, racionales y polizones (free rider) [26], estos últimos son individuos o entes que consumen más que una parte equitativa de un recurso, o no afrontan una parte justa del costo de su producción. El comportamiento de los distintos tipos de nodo es el siguiente: los altruistas siempre cooperan, los racionales cooperan una de cada dos veces y los polizones nunca cooperan. Los valores que toma la disponibilidad y el umbral de pago mínimo para cada nodo, está en función del tipo al que pertenece y se asignan como sigue:

Disponibilidad

La disponibilidad para los tres tipos de nodo está entre [1 %, 100 %]y se asigna utilizando una distribución uniforme.

Umbral de Pago

La asignación del umbral de pago a los nodos se realiza en función de la disponibilidad que tiene, utilizando una distribución uniforme.

- Nodos con disponibilidad en el intervalo [1 %, 34 %), se asigna umbral de pago entre [0.01, 0.34).
- Nodos con disponibilidad en el intervalo [34 %, 67 %), se asigna umbral de pago entre [0.34, 0.67).
- Nodos con disponibilidad en el intervalo [67 %, 100 %] se asigna umbral de pago entre [0.67, 1].

Es importante mencionar que la disponibilidad de los nodos, la cual es calculada con la ecuación 5.3, es multiplicada por 100 para obtener valores porcentuales de disponibilidad.

Generación de la estrategia

Para la generación de la estrategia de cada nodo, consideramos lo siguiente: en el caso de los nodos altruistas las estrategias consisten únicamente de "1", en el caso de los racionales consisten de 50 % de "0" y 50 % de "1" y en el caso de los polizones únicamente de "0". Por ejemplo, si la estrategia fuera de tamaño 6, entonces una que representa a un nodo altruista sería de la forma: 111111, la de un nodo racional podría ser por ejemplo: 110100 ó 001011 y la de un nodo polizón sería: 000000.

Poblaciones

Para la experimentación consideramos 5 poblaciones, con distintos porcentajes de cada tipo de nodo y que listamos a continuación:

- Población 1: 100 % de nodos polizones.
- Población 2: 98 % de nodos polizones y 2 % de nodos altruistas.
- Población 3: 98 % de nodos polizones y 2 % de nodos racionales.
- Población 4: 100 % de nodos racionales.
- Población 5: Para esta población consideramos la generación de las estrategias de los nodos de manera aleatoria, es decir, para generar cada elemento de una estrategia se genera un booleano, el cual utiliza un distribución uniforme. Si el valor del booleano es *true* entonces el valor del gen es 1, si es *false* el valor del gen es 0.

Las poblaciones 1-4, considerán el hecho de que la mayoría de los nodos en un sistema P2P se comportan de manera egoísta y que sólo una pequeña porción contribuye con espacio de almacenamiento [26]. La población 5 nos permite replicar el experimento realizado por Axelrod, en el sentido de considerar estrategias generadas de manera aleatoria.

Es importante señalar que no se consideró límite en el espacio de almacenamiento para los nodos.

6.2.2. Mecanismos de compensación

Se utilizaron los dos mecanismos de compensación presentados en la sección 5.4. Para el primer mecanismo los nodos acumulan la compensación económica otorgada por el documento, si este lo almacena. En el segundo mecanismo, los nodos obtienen una calificación de acuerdo al comportamiento que tuvieron con el sistema. Las consultas que los nodos podrán realizar, están en función de la cooperación mostrada.

6.2.3. Elección

Se consideran tres tipos de elección de nodos: secuencial, aleatoria y turno rotatorio (ver sección 5.5.2), cuya finalidad es la forma en cómo se eligen los nodos para duplicar un documento.

6.2.4. Documentos

Los documentos que buscan ser almacenados en el sistema cuentan con tres características: tamaño, pago máximo y disponibilidad requerida.

- El tamaño del documento (TD), indica cuánto mide el documento. Durante la experimentación el valor de TD es de 1.
- El pago máximo (PM), indica cuánto es el máximo que está dispuesto a pagar un documento por ser duplicado (pago que se da a cada nodo que lo almacena).
- La disponibilidad requerida por el documento (DRD), se refiere a cuán disponible desea estar el documento en el sistema.

Para la asignación de pago máximo y disponibilidad requerida, utilizamos una distribución uniforme entre [0.1,1].

6.2.5. Costo por consultas

Cuando se utiliza el mecanismo por compensación económica, los nodos deben pagar un costo por los documentos que consultan. Este costo estará en función del umbral de pago mínimo de los nodos, es decir el costo por consumir será igual al umbral de pago de los nodos.

6.2.6. Parámetros para el algoritmo genético

Evaluación: La aptitud de los nodos, es la razón de consultas exitosas la cual es calculada para cada uno de ellos utilizando la ecuación 5.2 que se muestra en la sección 5.5.1.

Selección: Utilizamos la técnica de escalamiento sigma, ya que buscamos que los nodos más cooperativos sean quienes produzcan descendientes, o bien se hagan copias de ellos para la siguiente generación.

Reproducción: Utilizamos cruza en un punto y mutación uniforme, con probabilidad de ocurrencia de 0.5 y 0.001 respectivamente. Se recomienda que el valor de probabilidad de cruce esté entre (0,1) [36] [35].

Generaciones: El número de generaciones fue de 1000 para todas las simulaciones realizadas.

Es importante mencionar que de cada simulación que se llevó a cabo se realizaron 30 repeticiones, con la finalidad de obtener un 95 % de confiabilidad de los datos obtenidos.

6.3. Experimentación

6.3.1. Mejor mecanismo de compensación: económica y de reconocimiento social

Objetivos

- Observar qué mecanismo de compensación influye en mayor medida en la evolución del comportamiento de los nodos de un sistema P2P, poniendo particular atención en la cantidad de documentos que se almacenan en el sistema, partiendo del hecho que la población es egoísta, es decir, no cooperante.
- Observar el impacto en la evolución de la cooperación de los nodos, al utilizar un determinado tipo de elección.

Parámetros

- Conjunto de 1500 documentos.
- Conjunto de 500 nodos.
- Costo por consulta para el mecanismo de compensación económica: x.
- Población: 1.

Resultados

De la figura 6.1, podemos observar los resultados obtenidos de aplicar los dos mecanismos de compensación y los tres tipos de elección. En las tres gráficas: *(A), (B) y (C)*, se observa que el mecanismo de reconocimiento social alcanza en un número menor de generaciones a almacenar el 100 % de los documentos, es decir, solo requiere poco más de 50 generaciones. El mecanismo de compensación ecónomica requiere de más de 300 generaciones para alcanzar el almacenamiento de casi la totalidad de los documentos generados.

Observamos también en las tres gráficas que el tipo de elección no influye en la cantidad de documentos que se almacenan en cada generación, ya que las curvas son similares en los tres casos, para los dos mecanismos.

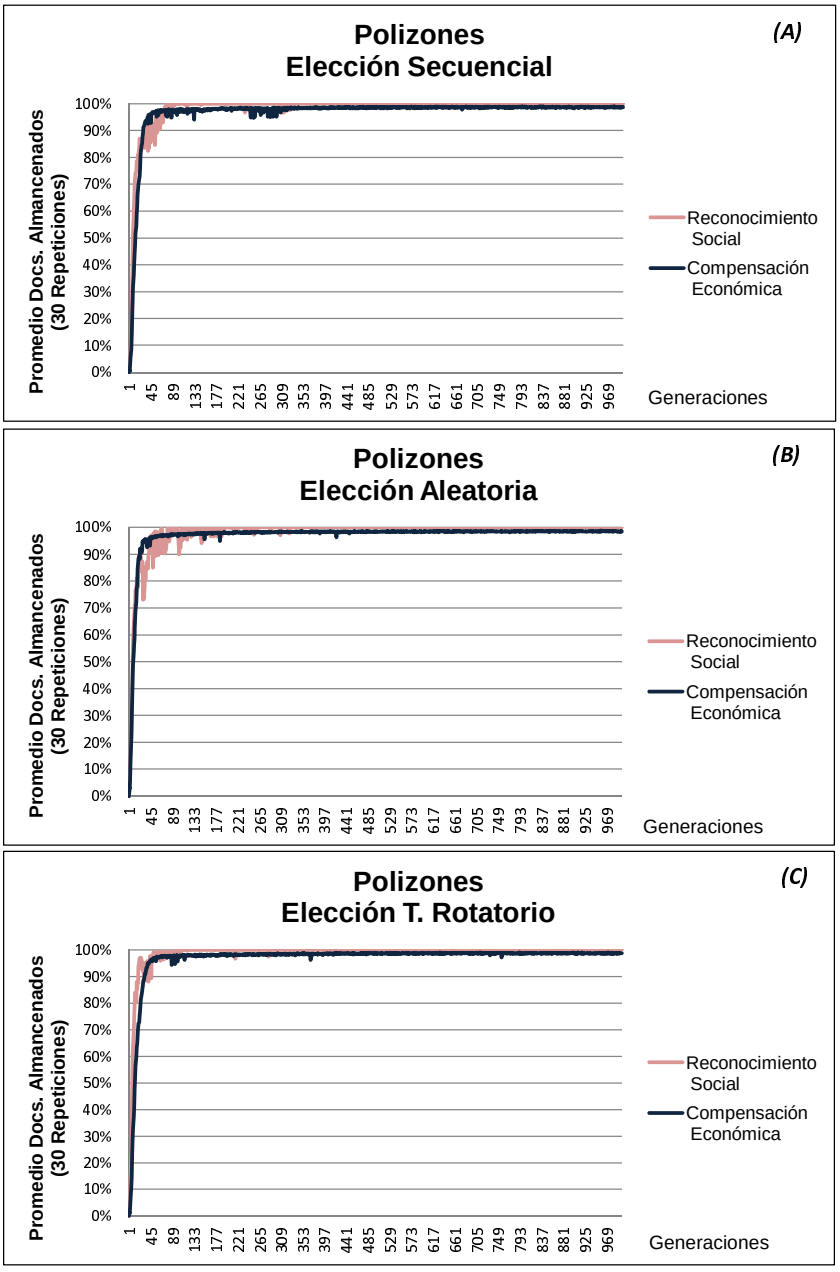

Figura 6.1: Promedio de documentos almacenados en cada generación, utilizando los dos mecanismos de compensación y elección (A) secuencial, (B) aleatoria y (C) turno rotatorio.

En las gráficas *(A), (B)* y *(C)* de la figura 6.2, se muestra la aptitud promedio obtenida por los nodos durante las mil generaciones. Recordemos que la aptitud de los individuos está en función de su cooperación, calculando el radio de documentos consultados exitosamente y que el máximo valor que puede alcanzar un nodo es 1.

Para el mecanismo de reconocimiento social, podemos observar que el tipo de elección sí afecta la evolución de la aptitud de los nodos, la cual va incrementando con el paso de las generaciones hasta alcanzar su máximo valor, haciéndolo en un número menor de generaciones, 450 aproximadamente, cuando se utiliza la elección aleatoria (ver figura 6.2 (B)). Para la elección secuencial (ver figura 6.2 (A))y turno rotatorio (ver figura 6.2 (C))se requieren aproximadamente 900 y 600 generaciones respectivamente. Deducimos que el tipo de elección aleatoria favorece la evolución de la aptitud, debido a que permite que todos los nodos tengan la misma oportunidad de ser seleccionados, a diferencia de los otros dos, lo cuales siguen un orden de elección. El hecho de que la aptitud promedio de los nodos alcance su máximo, nos dice que se obtiene la cooperación del 100 % de los nodos.

En el mecanismo de compensación económica podemos observar en las tres gráficas de la figura 6.2, que la aptitud de los individuos no supera el valor de 0.1, en ninguno de los tres tipos de elección, durante todas las generaciones, esto se debe a que si un documento alcanza la disponibilidad que requiere siendo almacenado por un solo nodo, el resto de los nodos no podrán acumular los recursos necesarios para realizar las consultas que desean, obteniendo así, un valor de aptitud de cero. Al promediar la aptitud de los nodos, ésta tendrá en la mayoría de los casos un valor cercano a 0. Es importante recordar que en este mecanismo sólo obtiene utilidad el nodo que duplica, a diferencia del mecanismo de reconocimiento social en donde los nodos obtienen utilidad únicamente por el hecho de declarar que están dispuestos a cooperar aún y cuando no lleven a cabo la duplicación.

Para el resto de la experimentación únicamente utilizaremos elección aleatoria, para dar la misma oportunidad a los nodos de ser electos para duplicar.

6.3.2. Composición de la población al final de las generaciones

Objetivo

Considerando distintos tipos de población inicial, observar el comportamiento al final de las generaciones, así como la constitución de la población.

Parámetros

- Conjunto de 1500 documentos.
- Conjunto de 500 nodos.
- Costo por consulta para el mecanismo de compensación económica: x.

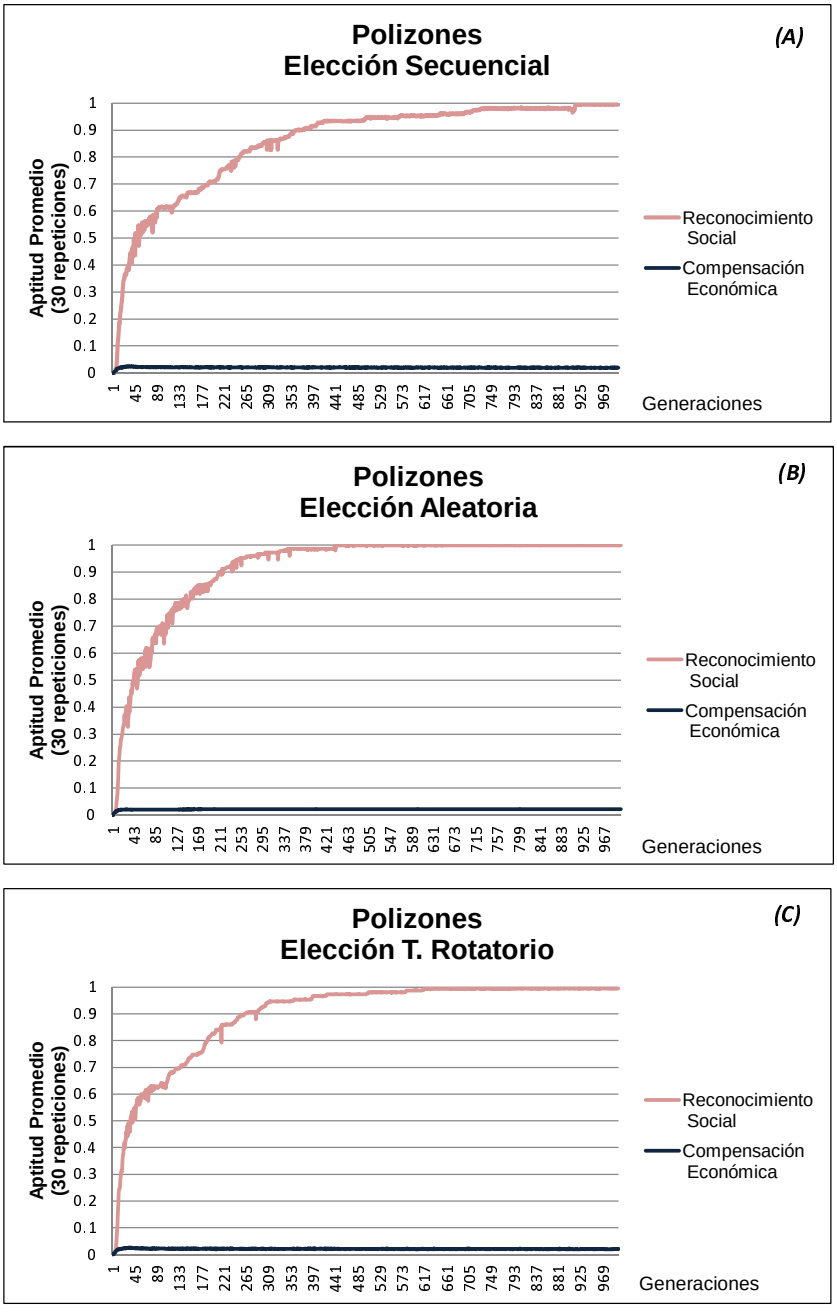

Figura 6.2: Aptitud promedio de los nodos en cada generación, utilizando los dos mecanismos de compensación y elección (A) secuencial, (B) aleatoria y (C) turno rotatorio.

- Poblaciones: se realizó una simulación para cada una de las siguientes poblaciones: 1, 2, 3, 4 y 5.

Resultados

Para describir los resultados obtenidos en este experimento, definimos dos tipos de comportamiento: programado y exhibido. El comportamiento programado, se refiere a la composición de las estrategias de los nodos, es decir, cuántos ceros y unos constituyen a la estrategia. El comportamiento exhibido, indica cuántas veces el nodo eligió cooperar (unos) o no cooperar (ceros) durante el juego, el cual se repite 1500 veces (# documentos) en cada generación.

Las figuras 6.3 y 6.4, muestran el resultado de analizar los datos de la población que jugó en la última generación de la simulación. Se utilizaron las cinco poblaciones iniciales, especificadas en la sección 6.2.1.

En las gráficas de la figura 6.3, observamos los resultados que se obtuvieron cuando se utilizó el mecanismo de reconocimiento social. En la grafica (A), podemos ver que la población 1, la cual inicialmente está constituida únicamente por nodos polizones, cuyas estrategias contienen únicamente ceros, al final de las generaciones la estrategias están constituidas por 27 % de bits en 1 y 73 % de bits en 0 (en promedio), ubicando a los nodos de la población entre polizones y racionales. La población 2, inicialmente compuesta por un 98 % de nodos polizones (estrategias constituidas por ceros) y un 2 % de altruistas (estrategias constituidas por unos), al final de las 1000 generaciones, está constituida por nodos cuyas estrategias se ubican entre racionales y altruistas, ya que contienen un 70 % de bits 1 y 30 % de bits en 0. La población 3 con un 98 % de nodos polizones y 2 % de nodos racionales al inicio de la simulacón, en la última generación, las estrategias de los nodos están constituidas en promedio por un 50 % de bits en 1 y un 50 % de bits en 0, lo que indica que los nodos tienden a ser racionales. La población 4 (inicial), cuyos nodos que la constituyen son racionales, es decir, cooperan una de cada dos veces, al final de la simulación, las estrategias de los nodos mantienen la proporción de ceros y unos, pues las estrategias contienen un 50 % de bits en 1 y un 50 % de bits en 0. Por último, la población 5, cuyas estrategias de los nodos son generadas de manera aleatoria (constituidas en promedio un 50 % de unos y 50 % de ceros), mantienen la misma cantidad de ceros y unos al término de las generaciones.

De lo anterior podemos observar que en cuanto al comportamiento programado, una población de polizones tenderá a cambiar su estrategia, de tal manera que se obtenga un beneficio como consecuencia (consultar documentos). También podemos observar que un porcentaje pequeño de nodos altruistas o racionales dentro una población egoísta (polizones), puede influenciar a cambiar las estrategias de la población completa, de tal manera que la población se encamine hacia el altruismo (población 2) y la racionalidad (población 3). Por otro lado una población integrada por nodos racionales se mantiene dentro de la racionalidad al final de la generaciones, al igual que la población cuyas estrategias son generadas de manera aleatoria.

Sin embargo, en la gráfica (B) de la figura 6.3, observamos que todas la poblaciones en la última generación exhiben un comportamiento casi altruista, es decir, de las 1500 oportunidades que tiene cada nodo para decidir cooperar o no con el sistema, más de 1475 (en promedio) deciden cooperar, aún y cuando ninguna población tenga nodos con estrategias altruistas.

Las gráficas de la figura 6.4 muestran los resultados de utilizar el mecanismo de compensación económica, los cuales muestran tendencias parecidas al utilizar el mecanismo de reconocimiento social. La diferencia radica en que con el mecanismo de compensación económica, se tiene una cantidad mayor de unos en todas las poblaciones con respecto al comportamiento programado (fig. 6.3 (A) y 6.4(A)), y en cuanto al comportamiento exhibido, podemos ver que, de las 1500 veces que cada nodo decide si coopera o no con la duplicación, más de 1445 veces decide cooperar (valor ligeramente menor que en el mecanismo de reconocimiento social).

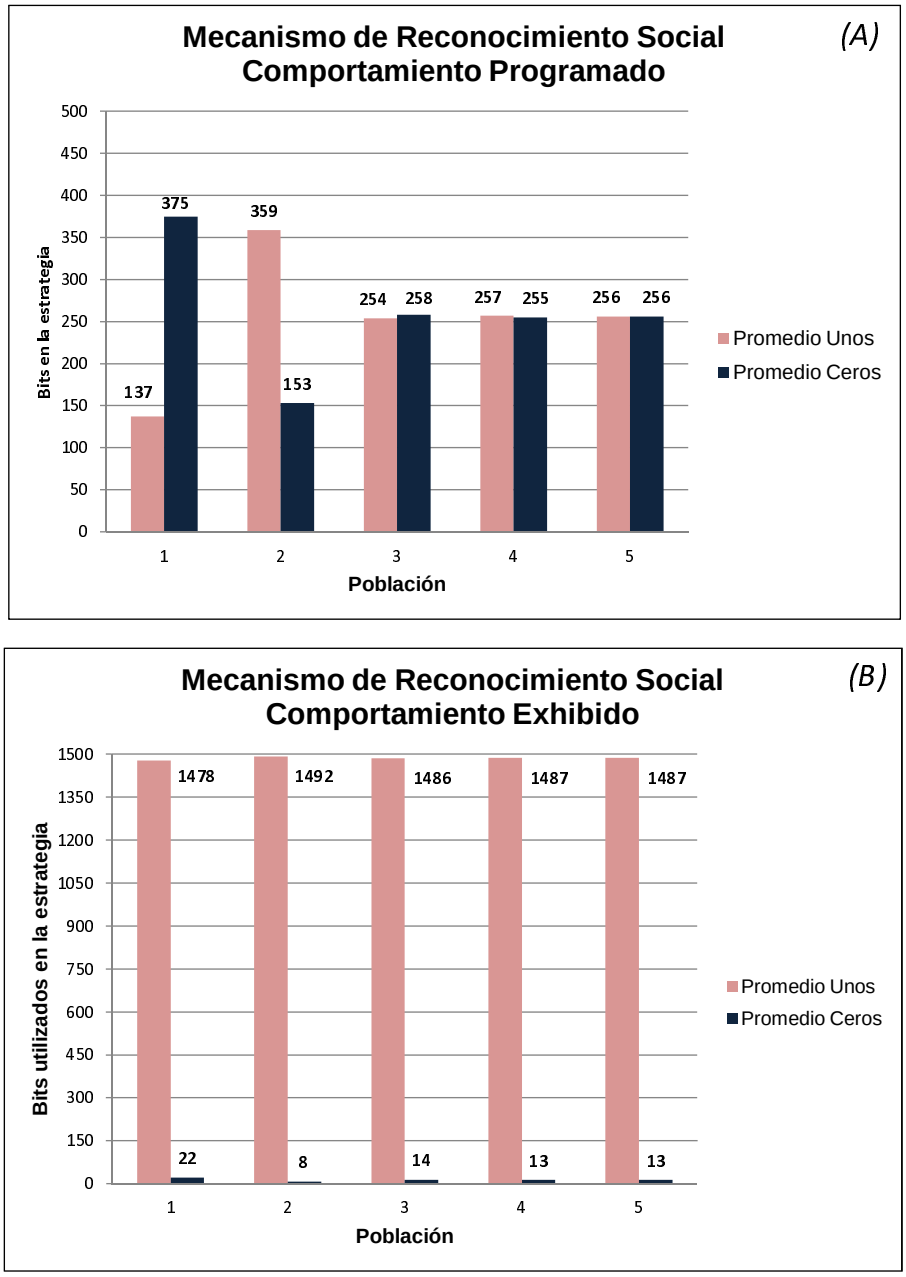

Figura 6.3: Comportamieto programado y exhibido de los nodos cuando se utiliza el mecanismo de reconocimiento social.

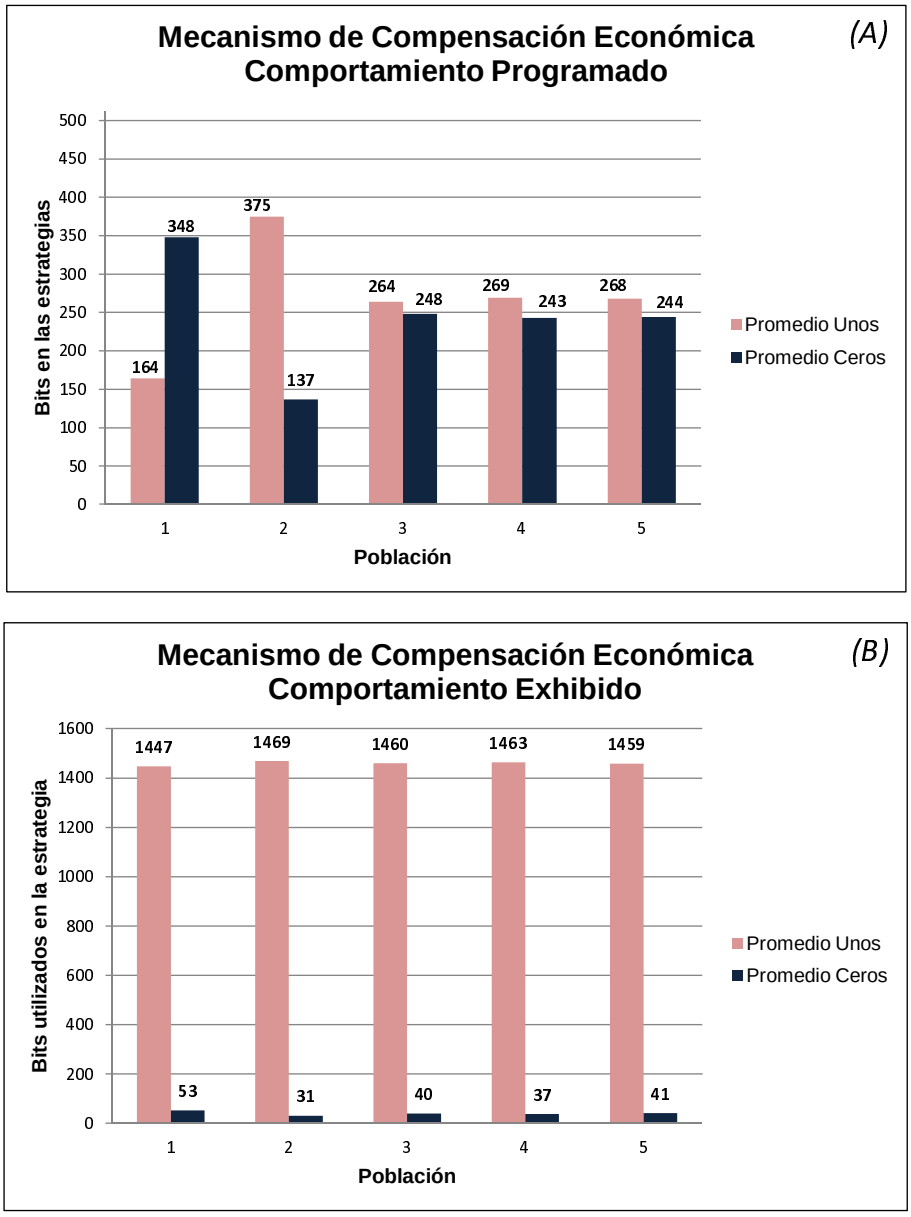

Figura 6.4: Comportamieto programado y exhibido de los nodos cuando se utiliza el mecanismo de compensación económica.

Capítulo 7
Conclusiones y trabajo futuro

El trabajo de investigación presentado, tuvo por objetivos estudiar y analizar como es afectada la evolución del comportamiento de los nodos de un sistema distribuido P2P, cuando se encuentran sometidos a la influencia de algún mecanismo que busca incentivar su cooperación (duplicar) con el sistema.

Para llevar a cabo lo antes descrito, se diseñó e implementó un juego, al que llamamos *Juego de duplicación,* en donde se modeló la situación estratégica de los nodos (duplicar, no duplicar) y el sistema, tiendo como finalidad que se lleve a cabo la duplicación de documentos. Además se consideró el hecho que la disponibilidad de los documentos está condicionada a la disponibilidad que presentan los nodos. El juego consiste en lo siguiente: un documento que requiere ser duplicado para alcanzar un determinado nivel de disponibilidad, solicita espacio de almacenamiento al sistema. Los nodos interesados se proponen para duplicar, si el documento alcanza la disponibilidad que requiere con los nodos candidatos, entonces se lleva acabo la elección de nodos que garanticen la disponibilidad del documento, para lo cuál se consideraron tres tipos de elección: secuencial, aleatoria y turno rotatorio. Una vez que el documento es almacenado, los nodos reciben una compensación por su cooperación.

También fue necesario diseñar dos mecanismos de compensación, para aplicarlos al juego de duplicación. El primero llamado, mecanismo de compensación económica, paga a los nodos que realizan la duplicación. El segundo denominado, mecanismo de compensación por reconocimiento social, los nodos son evaluados en función de la medida en la que cooperan con el sistema, aún y cuando no sean seleccionados para duplicar. Una vez que los nodos han decidido cuánto cooperar al sistema, podrán llevar a cabo la consulta de documentos en función del capital con que cuentan considerando al primer mecanismo, y la calificación obtenida en el segundo.

Diseñamos e implementamos un simulador, el cual nos permite observar la evolución del comportamiento de los nodos que juegan el juego de la duplicación, bajos los dos mecanismos de compensación propuestos. El simulador desarrollado hace uso de algoritmos genéticos para evolucionar el comportamiento de los nodos de la población.

Por último se llevó a cabo un conjunto de simulaciones que nos permitieron observar lo

siguiente:

- El mecanismo de reconocimiento social, consigue más rápidamente almacenar el 100 % de los documentos, que el mecanismo de compensación económica.
- La aptitud de los nodos si es afectada por el tipo de elección, ya que los nodos alcanzan su máximo más rápido cuando se utiliza elección aleatoria.
- Los dos mecanismos motivan a la población de un sistema P2P a cooperar, ya que aún y cuando la población de nodos cuyas estrategias no son altruistas (comportamiento programado), deciden en la mayoría de las veces cooperar (comportamiento exhibido).

Es importante mencionar que los resultados antes mencionados, consideran los escenarios descritos en las secciones 6.3.1 y 6.3.2.

Para trabajo futuro consideramos los siguientes puntos.

- Considerar el espacio de almacenamiento en los nodos para observar el impacto en los resultados obtenidos.
- Considerar distintas formas de evaluación de los nodos en el caso del mecanismo de compensación económica, ya que actualmente no capta completamente la cooperación de los nodos.
- Utilizar otras técnicas de cruza y mutación, así como variar la probabilidad de las mismas.
- Utilizar otras técnicas evolutivas, para observar qué resultados se obtienen.
- Considerar la topología del sistema P2P.
- Implementar el juego y los mecanismos de compensación en sistemas P2P simulados, para comparar si los resultados obtenidos se mantienen.
- Experimentar con las poblaciones finales obtenidas, para observar si se mantiene estables.
- Llevar a cabo paralelización, para disminuir tiempos de ejecución.

Apéndices

Apéndice A
Implementación del juego de duplicación, de los mecanismos de compensación y del algoritmo genético

En este capítulo describimos el desarrollo de un simulador, en el cual se implementó el juego de duplicación y de los mecanismos de compensación descritos en el capítulo anterior, así como la aplicación del algoritmo genético que nos permitirá observar los comportamientos que surgen en la población de nodos, cuando se encuentran bajo la influencia de los mecanismos de compensación.

A continuación describiremos las características principales del simulador.

A.1. Requerimientos

A.1.1. Funcionales

El simulador deberá ejecutar el juego descrito en el capítulo anterior, así como los mecanismos de compensación: económica y de reconocimiento social.

El simulador generará un conjunto de nodos con las características correspondientes (cromosoma, disponibilidad, umbral de pago mínimo) para cada uno, información que se almacenará en un archivo de texto, el cual será utilizado para leer la información de los nodos cuando se inicia la ejecución del juego.

El simulador generará el conjunto de documentos que solicitarán espacio de almacenamiento al sistema, el cual se almacenará en un archivo de texto, de donde los datos de cada documento serán leídos.

El simulador actualizará los resultados que obtiene cada nodo en cada interacción del juego y la situación del documento en el sistema, guardando al final de las iteraciones el dato de cuántos documentos se almacenaron en cada generación en un archivo de texto.

El simulador al final de cada iteracción del juego, ejecutará la evaluación, selección y reproducción, para lo cual necesitará la lista de nodos con todos sus datos actualizados.

El simulador almacenará al final de las generaciones el conjunto de nodos que juega en la última generación y los documentos almacenados en un archivo de texto.

A.1.2. No funcionales

Usabilidad

El simulador debe ser fácil de utilizar, evitando que el código sea modificado para actualizar alguno de los parámetros que se necesitan para su correcto funcionamiento, por lo cual se requiere que dichos parámetros sean especificados de manera correcta en un archivo de texto, de donde serán leídos por el simulador.

Rendimiento

El simulador deberá ejecutarse en un tiempo razonable, el cuál deberá estar en función del número de documentos, el número de nodos y el número de generaciones que se estén considerando en el juego.

Modularidad

El simulador deberá permitir ejecutar cualquier tipo de juego o problema en el simulador, para lo cual debe ser suficiente modificar el módulo que ejecuta el juego y el que genera la población inicial, ya que son propios del problema que se está estudiando. Los módulos que ejecutan la evaluación, selección y reproducción no necesitan ser modificados.

A.2. Arquitectura

A.2.1. Descripción general

En la figura A.1, podemos ver los componentes del simulador desarrollado, el cual consta de cinco módulos principales: coordinador, generador de población inicial, juego de duplicación, evaluación de la población, selección de nodos y reproducción. Algunos de estos módulos contienen submódulos que permiten llevar a cabo las tareas que deben realizar.

El simulador inicia su ejecución en el coordinador, el cual ejecuta cada uno de los demás módulos antes mencionados. En primer lugar ejecuta el generador de población inicial, el cual se encarga de generar los nodos requeridos, así como los documentos que se utilizarán en el juego.

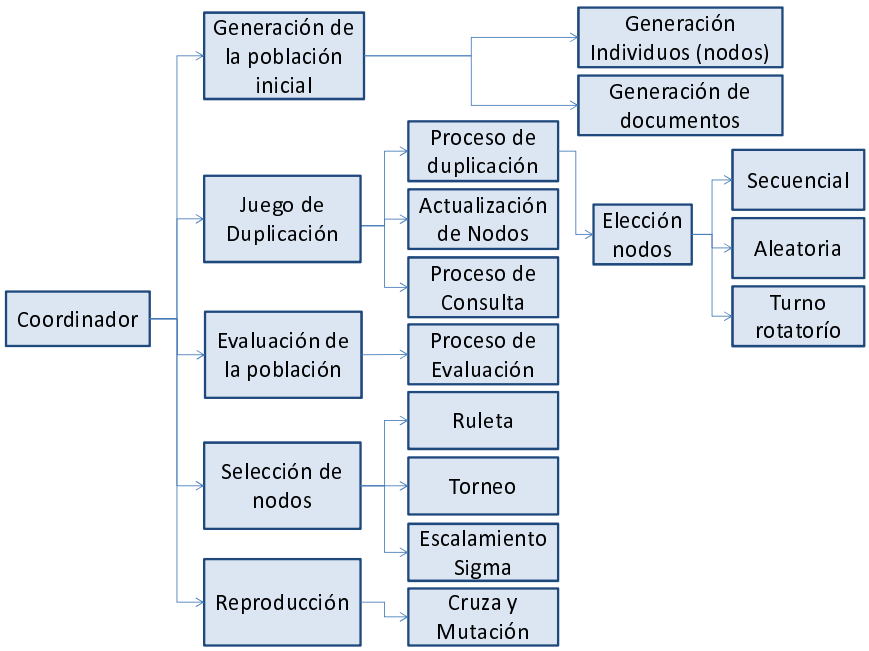

Figura A.1: Vista general del simulador.

Una vez que se tiene la población de nodos y el conjunto de documentos, se inicia la ejecución del juego el cual se itera en función del número de documentos generados. El juego consiste en correr el proceso de duplicación, la actualización de nodos y el proceso de consulta.

Una vez que todos los documentos generados han pasado por la ejecución del juego, se lleva a cabo el proceso de evaluación, selección y reproducción. A lo antes descrito se le considera una generación, repitiendo el proceso el número de generaciones ingresado, a excepción del módulo de generación de la población inicial.

A continuación describiremos con más detalle cada uno de los módulos del simulador.

A.2.2. Coordinador

Funcionalidad. Es el encargado de coordinar todos los componentes que se ejecutan en el simulador. El diagrama de secuencia que se muestra en la figura A.2 se muestra cómo se van ejecutando cada una de las partes del simulador.

Datos de entrada. La tabla A.1 muestra los datos de entrada que necesita el coordinador, los cuales son necesarios para iniciar la ejecución de cada uno de los módulos del simulador. Los datos son leídos de un archivo de texto y son utilizados durante toda la simulación.

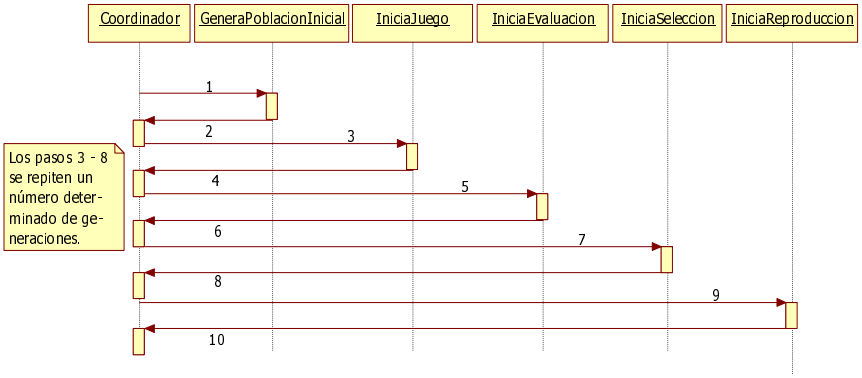

Figura A.2: Diagrama de secuencia de la ejecución del algoritmo genético.

Datos	Función
Tamaño de la estrategia	Cuántas posiciones se necesitan para almacenar el resultado de una interacción.
Tamaño de la historia	Tamaño de la historia que se va a considerar para el juego.
Número de nodos	Jugadores.
Número de generaciones	Repeticiones del juego.
Número de documentos por generación	Documentos que se consideran para cada repetición del juego.
Probabilidad de cruza	Probabilidad de que la cruza se lleve a cabo.
Probabilidad de mutación	Probabilidad de que la mutación se lleve a cabo.
Tipo de mecanismo	Mecanismo de compensación que se utilizará.
Tipo de elección	Tipo de elección a utilizar.
Tipo de selección	Tipo de selección a utilizar.
Cruza historia	Indica si la historia es considerada al momento de producirse la cruza de los cromosomas.
% de Población	Indica cómo está constituida la población.

Tabla A.1: Datos de entrada requeridos para la ejecución del simulador.

Datos de salida. Se genera un archivo con la población obtenida de la última generación.

Interacciones. El coordinador interactúa con los siguientes módulos:

- genera población inicial
- juego de duplicación
- evaluación de la población
- selección de nodos y reproducción

Ejecución. El módulo coordinador está en ejecución el tiempo total que dura la simulación, el cual está en función del número de generaciones. Además es el encargado de llamar a los demás modulos, de proveer los datos que necesitan y de recibir los resultados que generan.

A.2.3. Generador de población inicial

Funcionalidad. Encargado de generar el conjunto de cromosomas que representan las estrategias a utilizar por los nodos del sistema. El cromosoma se genera creando primero la parte que contiene la estrategia de los nodos. Una vez generadas todas las estrategias, se genera la historia ficticia para completar el cromosoma para cada uno de los nodos, así como las características de disponibilidad y pago mínimo requerido.

Datos de entrada. Los datos de entrada para este módulo son los siguientes:

- Número de nodos.
- Tamaño de la estrategia.
- Tamaño de la historia.
- Número de documentos.

Datos de salida. Genera la lista de nodos y de documentos que serán considerados en el juego.

Interacciones. Únicamente interactúa con el módulo coordinador.

Ejecución. Se ejecuta una sola vez, al inicio de la simulación.

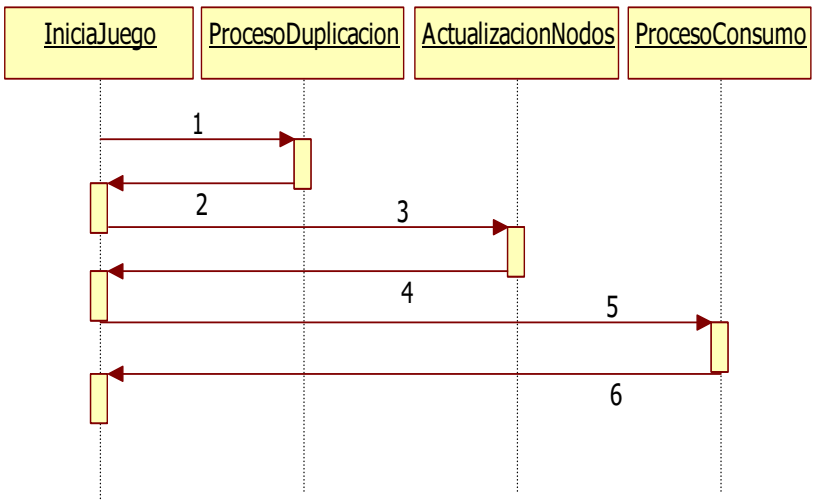

Figura A.3: Diagrama de secuencia del módulo de Evaluación de la Población.

A.2.4. Juego de duplicación

Funcionalidad. Encargado de ejecutar el proceso de duplicación, la actualización de nodos y el proceso de consulta, así como de registrar el número de documentos almacenados en cada iteración del juego. En el proceso de duplicación se lleva a cabo la toma de decisiones de los nodos, el cálculo de la disponibilidad que se provee a los documentos y la elección de los nodos en caso de que el documento obtenga la disponibilidad que necesita. En la actualización de nodos, se actualizan los parámetros de los nodos (dinero acumulado en el mecanismo de compensación económica y el número de cooperaciones en el de reconocimiento social), la situación de los documentos y el comportamiento del nodo. En el proceso de consulta, se elige de manera aleatoria un conjunto de documentos de aquellos que el sistema logró almacenar, para cada uno de los nodos. La figura A.3 muestra el diagrama de secuencia del módulo juego de duplicación.

Datos de entrada. El módulo de juego de duplicación, necesita los datos de entrada que se muestran en la tabla A.1, la lista de nodos y la lista de documentos. El proceso de duplicación requiere de la lista de nodos y los datos del documento que busca espacio de almacenamiento. Para la actualizacón de los nodos, se requiere de la lista de nodos, y los datos del documento. Para ejecutar el proceso de consulta, se necesitan la lista de nodos y la lista de objetos que el sistema logró almacenar.

Datos de salida. Se obtiene el número de documentos que se almacenaron, el cual que se guarda en un archivo de texto. Además devuelve al coordinador la lista de nodos actualizada.

Interacciones. El módulo interacciona con el coordinador, así como con el proceso de duplicación, la actualización de nodos y el proceso de consulta.

Ejecución. La ejecución del juego de duplicación termina cuando el proceso de consulta concluye. El proceso de duplicación y la actualización de los nodos se repite en función del número de documentos generados . Una vez que los dos módulos antes mencionados han terminado de ejecutarse, se llama al proceso de consulta.

A.2.5. Evaluación de la población

Funcionalidad. En este módulo se actualiza la razón de consultas exitosas para cada nodo, la cual se calcula utilizando la ecuación de utilidad que se muestra en la sección 5.5.1. El resultado que obtiene cada nodo se considera la aptitud de cada uno de ellos.

Datos de entrada. La evaluación de la población únicamente requiere como datos de entrada la lista de nodos actualizada.

Datos de salida. Se devuelve al coordinador la lista de nodos con su aptitud actualizada.

Interacciones. Este módulo interactúa únicamente con el coordinador.

Ejecución. Se ejecuta una vez en cada generación, cuando el módulo juego de duplicación a terminado.

A.2.6. Selección de nodos

Funcionalidad. En este módulo se lleva a cabo la selección de los nodos que generarán la población para la siguiente generación, el cual contiene los tres tipos de selección descritos en la sección 4.1.3.

Datos de entrada. Para su ejecución se requiere de lista de nodos y los datos de la tabla A.1.

Datos de salida. Genera un lista que contiene las parejas seleccionadas, la cuál será utilizada para llevar a cabo la reproducción.

Interacciones. Únicamente interactúa con el coordinador.

Ejecución. Se ejecuta una vez en cada generación, una vez que el módulo de evaluación de la población a terminado.

A.2.7. Reproducción

Funcionalidad. En este módulo se lleva a cabo la cruza de las parejas generadas en el módulo de selección de la población, y la mutación sobre la población resultante de la cruza.

Datos de entrada. Para la ejecución de este módulo se necesitan como datos de entrada la lista de parejas seleccionadas, y el parámetro que indica cual es la probabilidad de cruza y de mutación.

Datos de salida. Al final de la reproducción se genera una nueva lista con los nodos resultantes de la reproducción.

Interacciones. Únicamente interactúa con el coordinador.

Ejecución. Se ejecuta una vez en cada generación, una vez que el módulo de selección de los nodos a terminado.

A.3. Plataforma de desarrollo

La implementación del simulador se realizó utilizando el lenguaje de programación orientado a objetos, java 1.6 [43].

Como entorno de desarrollo se utilizó Eclipse SDK, versión: 3.1.2, Copyright © 2011 Fundación de Software Apache [44].

Al ser java un lenguaje de programación multiplataforma, la programación del simulador y la ejecución de las simulaciones se llevo a cabo utilizando los sistemas operativos Linux (Ubuntu) y Windows.

Referencias

[1] G. Coulouris , J. Dellimore y T. Kindberg. *Sistemas Distribuidos: Conceptos y Diseño.* 2da Edición, Addison-Wesley 1994, capítulo 1.

[2] S. Androutsellis-Theotokis y D. Pinellisa. "Survey of Peer-to-Peer Content Distribution Technologies." *ACM Computing Surveys,* vol. 36, pág. 335-371, Dic. 2004.

[3] Napster. "Página de Napster." Internet: http://music.napster.com/, [May. 2011].

[4] OceanStore. "Página del proyecto OcenaStore." Internet: http://oceanstore.cs.berkeley.edu/. [May. 2011].

[5] M. Placek y R. Buyya. "Taxonomy of Distributed Storage Systems." *Reporte técnico,* Universidad de Melbourne, Laboratorio de sistemas distribuidos y cómputo grid. Jul. 2006.

[6] Genoma@Home. "Página del proyecto Genome@Home." Internet: http://genomeathome.stanford.edu/. [May. 2011].

[7] Seti@Home. "Página del proyecto Seti@Home." Internet: http://setiathome.berkeley.edu/. [May. 2011].

[8] BitTorrent. "Página del proyecto BitTorrent." Internet: http://www.bittorrent.com/intl/es/. [Oct. 2011].

[9] Freenet. "Página del proyecto Freenet." Internet: http://freenetproject.org/index.html. [Oct. 2011].

[10] J. Gray y D.P. Siewiorek. "High-Availability Computer Systems." *IEEE Computer.* vol. 24, Sep. 1991

[11] V. Gopalakrishnan, B. Silaghi, B. Bhattacharjee, y P. Keleher. "Adaptive Replication in Peer-to-Peer Systems." En actas de *24th International Conference on Distributed Computing Systems.*, 2004, pág.360-369.

[12] R. Bhagwan, D. Moore, S. Savage, y G. M. Voelker. "Replication Strategies for Highly Available Peer-to-Peer Storage Systems." *Future directions in distributed computing: research and position papers*, Springer-Verlag, Berlin, Heidelberg, 2003.

[13] A. G. Lazalde Cruz. *Evaluación de técnicas de redundancia en Sistemas de Almacenamiento P2P*. Tesis de Maestría, UAM - Iztapalapa, México D.F., 2009.

[14] G. Nychis, A. Andreou, D. Chheda, A. Giamas. *Analysis of Erasure Coding in a Peer-to-Peer Backup System*. Information Networking Institute, Carnegie Mellon University.

[15] J. Tian, Z. Yang, Y. Dai. "SEC: A Practical Secure Erasure Coding Scheme for Peer-to-Peer Storage System". En actas de *14th Symposium on Storage System and Technology*, 2008.

[16] D. Geels, J. Kubiatowicz."Replica Management Should Be a Game". En actas de *10th workshop on ACM SIGOPS European workshop,* Saint-Emilion, France, Jul. 2002.

[17] C. Schmidt. *Game Theory and Economic Analysis*. Matter and Selection, 2002.

[18] Robert Gibbons. *Un Primer Curso de Teoría de Juegos*. Universidad de Cornell, 1992.

[19] J. N. Webb. *Game Theory: Decision, Interaction and Evolution*. Editorial Springer, 1988.

[20] J. Pérez, JL. Jimeno y E. Cerda. *Teoría de Juegos*. España: Pearson Prentice-All, 2004.

[21] C. Buragohain, D. Agrawal y S.Suri. "A Game Theoretic Framework for Incentives in P2P Systems." En actas de *The 3rd International Conference on Peer-to-Peer Computing,* p. 48 , Sep. 2003.

[22] B. G. Chun, K. Chaunduri, H. Wee, M. Barreno, C. H. Papadimitriou, J. Kubiatowicz. "Selfish Caching in Distributed Systems: A Game-Theoretic Analysis." En actas The twenty-third annual ACM symposium on Principles of distributed computing, pág. 21-30, Jul. 2004.

[23] N. Laoutaris, O. Telelis, V. Zissimopoulos y I. Stavrakakis. "Distributed Selfish Replication." *IEEE Transactions on Parallel and Distributed Systems,* vol. 17, pág. 1401-1413, Dic. 2006.

[24] S. U. Khan y I. Ahmad. "A Pure Nash Equilibrium based Game Theoretical Method for Data Replication across Multiple Servers." *IEEE Transactions on Knowledge and Data Engineering,* vol. 20, pág. 346- 360, Abr. 2009.

[25] http://plato.stanford.edu/entries/game-evolutionary/

[26] K. Zhang, N. Antonopoulos y Z. Mahmood. "A Review of Incentive Mechanisms in Peer-to-Peer Systems." *First Internactional Conference on Advances in P2P Systems*, Oct. 2009.

[27] B. Yang y H. Garcia-Molina. "PPay: micropayments for peer-to-peer systemas." En actas de *10th ACM conference on Computer and Communications Security.* Washington D.C. USA, 2003.

[28] "Sitio web de PayPal." Internet: https://www.paypal.com/ [fecha]

[29] "Sitio web DC++." Internet: http://www.dcplusplus.sourceforge.net/ [fecha]

[30] S.D.Kamvar, M.T. Schlosser, y H. Garcia-Molina. "The Eigentrust algorithm for reputation management in P2P networks." En actas de *12th International Conference on World Wide Web.* Budapest, Hungria: ACM, 2003.

[31] Z. Runfang y H. Kai. "PowerTrust: A Robust and Scalable Reputation Systems for Trusted Peer-to-Peer Computing." IEEE Trans. Parallel Distributed Systems. vol 18, pág. 460-463, 2007.

[32] X. Li y L. Liu. "PeerTrust: Supporting Reputation-Based Trust for Peer-to-Peer Electronic Communities. *IEEE Trans, and Knowloge and Data Eng.* vol. 16, pág. 843-857, 2004.

[33] T. Bocek, M. Shann, D. Haussher y B. Stiller. " Game Theoretical analysis of incentives for large-scale, fully decentralized collaboration networks. *,Ä*

[34] R.T.B. Ma, S.C.M. Lee, J.C.S. Lui y D.K.Y. Yau. "Incentive and service differentiation in P2P networks: a game theoric approach." *IEEE/ACM Trans. Network.* vol. 14, pág. 978-991, 2006.

[35] A.P. Alves da Silva y D.M. Falcão. "Fundamentals of Genetic Algorithms". Modern Heuristic Optimization Techniques: Theory and Applications to Power Systems, capítulo 2, K. Y. Lee y M. A. El-Sharkawi, Ed. John Wiley Sons, 2008.

[36] A. León. *Diseño e implementación en hardware de una algoritmo bio-inspirado.* Tesis de Maestría, IPN - CIC, México D.F., 2009.

[37] R. Axelrod. "Evolving new strategies: The evolution of strategies in the iterated prisoner's dilemma." *In Axelrod, R (Ed.): The Complexity of Cooperation: Agent-Based Models of Competition and Collaboration,* Princenton, pág. 10-32, 1997.

[38] R. Axelrod y W. D. Hamilton. "The Evolution of Cooperation." *Science New Series,* vol. 211, pág. 1390-1396, Mar. 1981.

[39] C. A. Coello Coello- "Introducción a la Computación Evolutiva". Internet: http://delta.cs.cinvestav.mx/ccoello/compevol/apuntes.pdf. May. 2001 [Ago. 2011].

[40] K. A. De Jong. *An Analysis of the Behavior of a Class of Genetic Adaptative Systems.* Tesis de Doctorado, Universidad de Michigan, 1975.

[41] T. Weise. "Global Optimization Algorithms - Theory and Aplications." e-book, segunda edición, 2009, capítulo 2, pág. 95 - 115.

[42] T. Wang, A. Nakao, A. V. Vasilakos y J. Ma. "Overview of modeling and analysis of incentive mechanisms based on evolutionary game theory in autonomous networks." *10th International Symposium on Autonomous Decentralized Systems*, Mar. 2011.

[43] "Página de Oracle." Internet: http://www.oracle.com/us/technologies/java/index.html. [Oct. 2011].

[44] "Sitio web de Eclipse." Internet: http://www.eclipse.org. [Oct. 2011].