MANAGING IDENTITY IN P2P SYSTEMS BASED ON DHT

MANEJO DE IDENTIDADES EN SISTEMAS P2P BASADO EN DHT

Recibido: 3 de julio 2015- aceptado: 16 de septiembre 2015

Ricardo Villanueva[1]
Universidad de Londres

Keywords:
P2P, networks, information, connections, communications

Abstract

This paper contains information relating to what concerns the networks P2P, there was analyzed the way in which the nodes relate between yes and of which it forms the organization in which they are distributed on having entered to the network. Every node on having created a network or to join the already existing one possesses an identifier which gives origin to the way in which there are distributed the following nodes that will join the network, The fault takes root in that one of the nodes already connected to the existing network they can be malicious and in originating points of assault to the network affecting the confidentiality of the information distributed between other nodes of the network or to modify the routing of the information supplied across the cap of application, since these nodes only with the fact of being in the network are responsible for the communication that is realized between certain nodes located in the ring. Were indicated detailed the processes of connection, communication and stabilization of the nodes by means of the simulation of the networks P2P in overlay weaver, showing I obtain the characteristics and results of the simulation.

Palabras clave:
P2P, redes, información, conexiones, comunicacion

Resumen

Este artículo presenta las redes P2P, se analiza la manera en la cual los nodos se relacionan entre sí y de que forma en la cual se distribuyen al entrar a la red. Cada nodo al crear una red o unirse a una ya existente posee un identificador el cual da origen a la manera en la que se distribuyen los siguientes nodos que se unirán a la red, la falla radica en que uno de los nodos ya enlazados a la red existente pueden ser maliciosos y originar puntos de ataque a la red afectando la confidencialidad de la información distribuida entre los demás nodos de la red o modificar el enrutamiento de la información suministrada a través de la capa de aplicación, ya que estos nodos solo con el hecho de estar en la red son responsables de la comunicación que se realiza entre ciertos nodos localizados en el anillo. Se indicaran detalladamente los procesos de conexión, comunicación y estabilización de los nodos por medio de la simulación de las redes P2P en overlay weaver, mostrando consigo las características y resultados de la simulación.

1. Royal Hollaway, University of London, London, England. E-mail: Ricardo.VillanuevaPolanco.2013@live.rhul.ac.uk
*Este artículo es asociado al artículo de investigación y tecnológica. Manejo de Identidad en Sistemas P2P basado en DHT

I. INTRODUCTION

Al hablar sobre el uso de Internet, estamos hablando de la necesidad de usar una red, que nos permita enrrutar paquetes y poder comunicarnos con otros usuarios, igualmente poder referenciar el comportamiento de dichos usuarios, saber dónde están, en qué estado están. Por otro lado la credibilidad de la información que es suministrada por éstos usuarios lo que nos brindara confiabilidad a nivel de comunicación.
Los identificadores en la red son los primeros pasos de seguridad en nuestra red si el proceso de identificación es malo cualquier usuario accedería a nuestra red, sin restricción alguna. Además mostraremos la importancia que posee el proceso de identificación y las consecuencias que traería consigo a la red P2P y cómo afectaría el rendimiento en nuestra red.

Mostraremos ataques de identidades como la asignación arbitraria de identificadores y el ataque sybil. Los cuales significan el comienzo de un mal funcionamiento e inseguridad de nuestra red.

La capa de enrutamiento supone gran importancia en una red Peer To Peer, debido que esta se encarga de llevar nuestros paquetes hacia los destinos deseados, por este motivo, esta capa soporta a sus niveles superiores, tales como la aplicación; es decir, que si queremos obtener información veraz e integra, se debe garantizar un buen funcionamiento de dicha capa.

En este trabajo tendremos la posibilidad de comprender el comportamiento generado por las fallas en la capa de enrutamiento y las relaciones con los ataques de identidades. Además de estudiar las causas por las cuales se genera un mal funcionamiento de la capa de enrutamiento, además de explicar la perdida de rendimiento de una red Peer To Peer por dichos ataques.

II. DESARROLLO DEL ARTÍCULO

REDES P2P

Los sistemas P2P son un nivel de aplicaciones de redes virtuales que tiene superposiciones de topologías propias y protocolos de enrutamiento. La topología de cada red define como los nodos están conectados el uno con el otro, mientras que el protocolo de enrutamiento define cómo los nodos pueden intercambiar mensajes con el fin de compartir información y recursos.

La topología de red y el protocolo de enrutamiento asociados a los sistemas P2P tienen una influencia significativa en las propiedades de aplicación tales como el rendimiento, la escalabilidad y fiabilidad del sistema.
Las topologías de redes P2P se pueden clasificar en dos categorías principales: estructurados y no estructurados, basados en su estructura.
Por "estructura" nos referimos al control sobre la creación de superposición y la ubicación de datos en la red.

REDES P2P ESTRUCTURADAS

En un intento por remediar el problema de escalabilidad de los sistemas no estructurados, algunos trabajos se han centrado en la introducción de la "estructura" o en las topologías de red.
La topología está estrechamente controlada, y el contenido puede ser distribuido de acuerdo a reglas específicas. Estas obras llevaron a la tercera generación de sistemas P2P (3GP), es decir, los sistemas P2P estructurados. Con el objetivo, básicamente de actuar como un índice de descentralizaciones estructuradas, para proporcionar una asignación entre el contenido (identificador de archivo, por ejemplo) y la ubicación (por ejemplo, el nodo dirección), en forma de una tabla de enrutamiento distribuido.
Las Redes estructuradas consisten en la división de un espacio clave entre compañeros, por lo que cada nodo es responsable de una ubicación de su espacio de claves específicas, es decir, debe guardar todos los recursos (o punteros) para esto se utilizan tablas de enrutamiento basadas en funciones hash **DHT** [1] [2].
Se dice que es un sistema estructurado porque los nodos de su red pueden producir una estimación (no con certeza) de que nodo es el más probable para almacenar ciertos datos. Ellos utilizan un modo de enfoque de la cadena de propagación, donde cada nodo tiene una

Structure	Decentralization		
	Hybrid	*Partial*	*Full*
Unstructured	Napster Publius	KaZaa Morpheus Gnutella2 Edutella	Gnutella FreeHaven
Structured Infrastructures			Chord CAN Trapestry Pastry
Structured Systems			OceanStore Mnemosyne Scan, Past Kademlia

decisión local sobre qué nodo es el indicado para enviar o atender una solicitud.

Tabla 1: clasificación de sistemas P2P en infraestructuras basados en estructura de red y grado de descentralización. Tomado de [3].

SISTEMAS P2P BASADOS EN DHT

Un sistema p2p basado en DHT (Distributed Hash Table) es un sistema estructurado que utiliza a las funciones hash para la ubicación y localización de los nodos en una red. Un sistema DHT considera un espacio de claves de una longitud determinada que es dividida en N partes siendo N el número de nodos en la red. A cada nodo le pertenece una clave única dentro del conjunto. Las funciones hash son de gran ayuda para este tipo de redes pues al momento de ubicar un peer en la red proveen O(log N), como límite superior en la longitud de las tablas de enrutamiento.

Este tipo de redes tiene muchas características:

- Los Datos y los nodo comparten el mismo espacio de direcciones, ósea O(log N).
- Los nodos intermedios mantienen información de enrutamiento a los nodos destinos.
- Se encamina salto a salto hasta llegar a nuestro objetivo.
- Como son sistemas totalmente distribuidos, la entrada y salida de los nodos se maneja automáticamente sin la necesidad de una entidad central.

CHORD

Chord [4] es uno de los múltiples sistemas que usan una topología en forma de anillo. Chord usa una función hash para asignar Claves que identifiquen a los nodos de tal forma que el impacto de un nodo al entrar y salir del sistema sea mínimo.

Este esquema descentralizado esta balanceado debido a que a cada nodo se le asigna el mismo número de claves, es por esto que cada vez que el sistema cambia, el intercambio de claves entre nodos es mínimo. Los nodos en Chord se organizan siguiendo una topología en anillo, donde cada nodo está conectado con el siguiente nodo en el anillo y con sólo otros O (log N) nodos del sistema, siendo N el número de nodos del sistema.

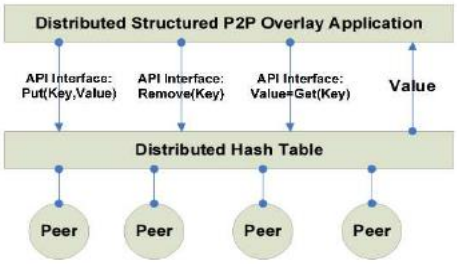


Fig. 2. Interfaz de una red P2P estructurada basada en DHT.

La función hash que se usa para generar las claves tanto para información como para nodos es SHA-1, la cual genera claves de 160 bits. Para el caso de los identificadores de los nodos, se aplica SHA-1 a la dirección IP del nodo, garantizando así que ningún otro nodo tendrá el mismo identificador.

En cuanto a la información almacenada en el sistema, para generar la clave se aplica la función hash a la propia información. Se ha elegido SHA-1 porque es necesario que la longitud de clave generada, m, por la función hash sea lo suficientemente grande para garantizar que no habrá colisiones, es decir distintas claves con el mismo identificador.

La tabla de enrutamiento de los nodos en Chord contiene un conjunto m (entradas), y el predecesor de este nodo. Asuma que el identificador de un nodo es n. La i-*ésima* entrada en la tabla entradas del nodo n, es el sucesor del identificador $n + 2^i - 1$. Se puede notar que la primera entrada en la tabla de cada nodo es su sucesor.

Este espacio de claves se mapea sobre un círculo módulo 2m. En este círculo, cada clave k es asignada al primer nodo cuyo identificador sea mayor o igual al suyo, dicho nodo se conoce como el sucesor (k) y representa el siguiente nodo en el círculo en sentido horario después de k. Cuando un nodo n entra en el sistema, algunas de las claves anteriormente asignadas al sucesor(n) son reasignadas a n. De la misma forma, cuando n deja el sistema, las claves de las que es responsable son reasignadas al sucesor(n). En la Figura 4, extraída de [1], podemos ver un ejemplo del funcionamiento de un sistema Chord: para un anillo con 10 nodos, el nodo N8 intenta encontrar la clave K54, para ello se va comunicando sucesivamente con sus vecinos hasta encontrar el nodo que posee K54.

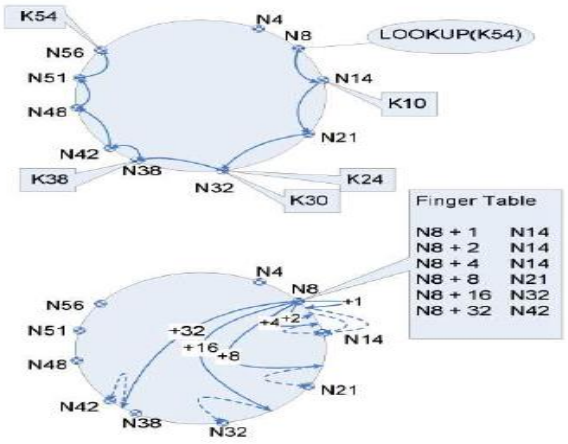


Fig. 4. Operación de búsqueda sobre un anillo Chord.

RED P2P FREEPASTRY DHT

Pastry [5] es realizado como soporte general para la construcción de una gran variedad de redes P2P. Aplicaciones de Internet como: el intercambio de archivos global, almacenamiento de archivos, la comunicación de grupo y los sistemas de denominación. Estas aplicaciones se han construido en la parte superior de pastry, incluyendo una utilidad global de almacenamiento persistente llamado PAST [6, 7] y una suscripción y publicación de grupos denominado SCRIBE [8].

Pastry es una red superpuesta de auto – organización de nodos, que identifica a cada nodo de forma única con una cadena de 128 bits [5]. Cuyo rango de identificación es de 0 hasta 2128-1.

En una red P2P como es la que Pastry crea, la información se almacena en tablas de hash. Por esto denominamos a estos sistemas DHT (Distributed Hash Table). Una tabla de Hash consiste en un array de elementos donde cada uno posee su correspondiente clave. De esta forma cuando añadimos un elemento a la tabla tenemos que calcular un índice a partir de la clave, intentando que sea lo más distinto posible al resto para mejorar la eficiencia en búsquedas futuras.

Cuando realizamos una búsqueda indicamos una clave con la cual, mediante el cálculo antes mencionado, obtendremos un índice válido. Con este índice la función encargada comparará en la tabla con los índices previamente insertados, encontrará el elemento asociado y nos lo devolverá.

Las tablas de Hash son muy efectivas cuando el número de entradas es alto.

Asumiendo una red basada en Pastry con N número de nodos, Pastry podría ir a cualquier nodo de la red en menos de log2bN pasos en promedios (b es un parámetro de configuración con el valor típico 4). Lo cual es de gran utilidad al momento de realizar una consulta en la red.

Kademlia.

Kademlia [9], implementada en los sistemas de intercambio de archivos Emule y Bitorrent, sigue la misma táctica que los anteriores de asignar a cada nodo un identificador de 160 bits contenido dentro del espacio de claves. Usa un sistema de enrutado orientado al identificador del nodo que está basado en el reenvió de mensajes al vecino que esté más próximo. La diferencia con el resto de DHTs es la métrica usada para calcular la distancia entre dos puntos del sistema. Kademlia usa una función XOR para calcular esta distancia debido a que esta función es simétrica. Kademlia se aprovecha de esta propiedad para aprender sobre las rutas en cada operación de búsqueda, ya que la función XOR asegura que todas las rutas que van hacia cierto punto del sistema terminan convergiendo.

Estilos de enrutamiento tradicionales

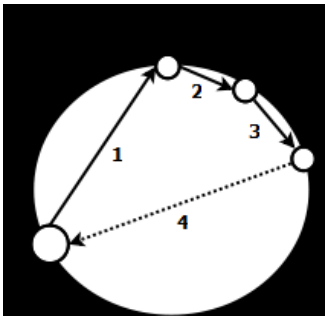
Los mecanismos de enrutamiento de las redes P2P de un proceso clave como lo es el mantenimiento a las tablas de enrutamiento para que el envio de mensajes dentro de la red sea eficientemente en todo momento. Existen tres estilos de implementación para el envío de mensajes: recursivo, iterativo y tracer.

En el enrutamiento recursivo, un nodo N, utilizando su tabla de enrutamiento, envía una consulta K a un nodo intermedio Y, en este proceso Y verifica si es el responsable de la consulta K. Si la respuesta es negativa, se repite el proceso anterior. Cuando la solicitud llega al nodo responsable por la consulta K, digamos R, él puede enviar su respuesta directamente al nodo iniciador de la consulta o usar el camino inverso de la petición. Ver figura1(a). Usando el enrutamiento recursivo el nodo iniciador no tienen ninguna opción de control sobre el proceso de enrutamiento y queda a disposición de los nodos intermedios, y por tanto, deteccion de nodos maliciosos no puede realizarse fácilmente. Trabajos que usan este estilo son Pastry [5], Chord [4] y Kademlia [6].

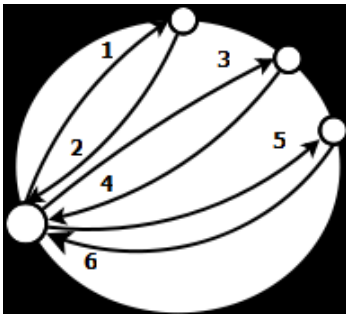
A diferencia del proceso de enrutamiento recursivo, en el proceso de enrutamiento iterativo, cada nodo intermedio Y, envía de vuelta al nodo N (iniciador) , la dirección IP del siguiente salto; Ver Figura 1(b).

Bajo este proceso de enrutamiento, N tiene control total del proceso. De esta manera, puede detectar nodos maliciosos. Algunas soluciones que utilizan este estilo son SeChord [11] y Myrmic [12].

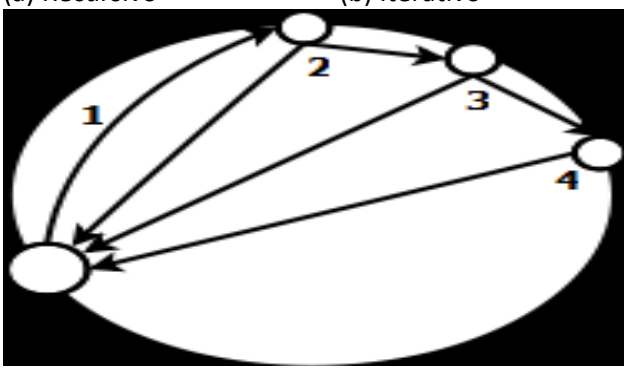
Finalmente, el enrutamiento tracer es una combinación de los estilos *recursivo* e *iterativo* [13]. Aquí cada nodo intermedio Y envía dos mensajes, uno al siguiente salto con la solicitud y otro, al iniciador N, con la información acerca del siguiente salto. Ver Figura 1(c). Este estilo provee cierto conocimiento al iniciador, pero no control total.



(a) Recursivo



(b) Iterativo



(c) Tracer

Figura 1: Estilos de enrutamiento [14]

ATAQUES A LA CAPA DE ENRUTAMIENTO

La capa de enrutamiento supone un eje importante en la estructura de una red cualquiera a la hora de compartir información, está se encarga de comunicarme con los demás clientes de la red y un mal funcionamiento de la capa de enrutamiento traería consigo ciertas consecuencias desde la redirección a un lugar donde no quiero estar o el compartir información con clientes de mi red que no quiero.

ATAQUES DE IDENTIDAD

Asignación de identificadores.

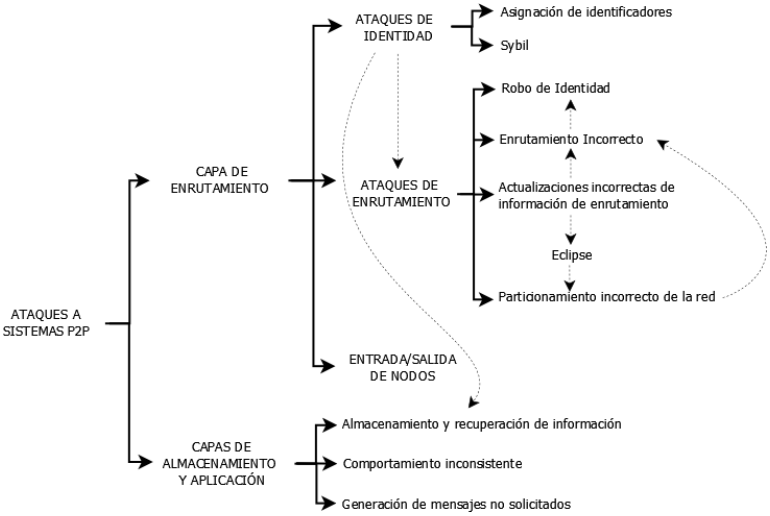
Este ataque ocurre cuando un nodo puede escoger su respectiva identificación en la red. Si esto ocurre el nodo puede ubicarse de forma incorrecta en cualquier parte de la red. El nodo atacante puede realizar un estudio de cuanto tráfico de información pasa por un segmento de la red p2p y ubicarse en el punto que el desee de dicho segmento esto daría como resultado el robo de información o la perdida de la misma.

Ataque Sybil.

Aun cuando exista una mejora en cuanto a la asignación segura de identificadores, esto no asegura que un nodo no pueda obtener múltiples posiciones en la red de forma arbitraria. Si esto sucede, esté nodo tomaría control sobre una gran parte de la red P2P y de esta manera inducir cualquiera ataque según su intensión o propósito. En particular, si no existen mecanismos para limitar el número de identidades por nodo, un atacante con acceso a millones de computadores puede desestabilizar la red y por ende a un numero de nodos en ella.

BOTNETS

Las redes Zombie o Botnets [10], son redes formadas por un conjunto de dispositivos electrónicos, en su mayoría computadoras, las cuales son infectadas con algunas instrucciones maliciosas para luego ser controladas de forma remota por atacantes, sin el conocimiento de su dueño; normalmente esta redes son utilizadas para muchas funciones Lo más frecuente es que una botnet se utilice para enviar spam a direcciones de correo electrónico, para la descarga de ficheros que ocupan gran espacio y consumen gran ancho de banda o para realizar ataques de tipo DDoS (*Distributed Denial Of Service*).



Clasificación y relación de los ataques en sistemas P2P

Ataques de enrutamiento

Los sistemas P2P estructurados como Chord [1], Pastry [5], utilizan el mismo principio durante el proceso de enrutamiento: cuando un nodo N recibe una solicitud de consulta, si N no posee el resultado inmediato de la

consulta, busca en su tabla de enrutamiento algún nodo cercano a la respuesta y reenvía la solicitud a ese nodo cercano y si este tampoco posee la respuesta reenvía a un nodo N más cercano a la respuesta hasta encontrar la repuesta. Si el sistema es confiable el nodo responsable de la consulta seguirá el mismo camino de esta, para responder a la consulta, si la capa de enrutamiento no funcionara correctamente podría tener las siguientes consecuencias:

1. el nodo responsable de la consulta nunca le llegara la petición.
2. un nodo malicioso podría suplantar al nodo responsable de la consulta, si esto ocurriese, el contenido de la información no sería confiable, pues el nodo malicioso podría cambiar la información en el camino.
3. otra opción seria, que la respuesta a la consulta no seguiría el protocolo de enrutamiento, hasta su destino final correcto, y podría ser enviada a un nodo que quiera apoderarse de dicha información y este enrutarla o no al nodo de dueño de la respuesta o de la petición.

Particionamiento incorrecto de la red.

Este tipo de ataques se presenta cuando un nodo N quiere ingresar a una red, para esto necesitaría enlazarse con un nodo ya existente (Bootstrap) en la red para luego unirlo a ella, si ese (Boostrap) es un nodo malicioso, localizado de forma ilegítima en nuestra red P2P podría unirnos a una red diferente formada de nodos maliciosos en la cual nos veremos completamente afectados.

Una solución a este tipo de ataques es que un nuevo nodo mantenga información sobre nodos honestos que conoce con anterioridad y use alguno de estos nodos como Bootstrap.

Denegación de Servicio Distribuida (DDoS)

Quizás los responsables de seguridad en sistemas informáticos por mucho que se empeñan en advertir de los peligros que se están asumiendo, los usuarios piensan que esas cosas sólo le ocurren a los demás, por lo que no se realiza esfuerzo alguno ni se adoptan las medidas recomendadas. Eso sí, el día que el usuario descubre que su sistema está siendo atacado, o incluso que ha sido atacado con éxito, mira a todas partes buscando un responsable. Pero nunca, o casi nunca, se mirará a sí mismo. Por esta razón muchas veces son víctimas de una denegación de servicio distribuida (DDOS) lo que traería como consecuencias la ruptura de servicio en la red a través de la interrupción de los componentes físicos de la red, el consumo de recursos de la red, el almacenamiento, los recursos de computación, o los recursos de ancho de banda, obstrucción de las comunicaciones y la interferencia en el estado de la información. Por ejemplo un DDoS attacker puede utilizar un Malware para mantener a un usuario fuera en los tiempos máximos de ejecución de CPU, o que se bloque el sistema mediante la activación de los errores en las ejecuciones de instrucciones.

Ataque de entrada/salida de nodos

En las redes P2P un nodo entra participa de la red y puede salir o entrar, si sale de la red debe existir un proceso de actualización de tablas de enrutamiento y volver a reasignar responsabilidades en la red, el mismo proceso pasa cuando un nodo entra a la red. Esto con el objetivo que el proceso de búsqueda y consulta funcione correctamente; un nodo malicioso puede hacer que el sistema colapse mediante este tipo de ataques, nos preguntamos ¿Cómo?, pues qué tal si un nodo malicioso puede engañar al sistema y hacerlo creer que sale de esté, obligando al sistema a re-balancearse de forma innecesaria causando exceso de tráfico y datos. Como consecuencia el desempeño y el rendimiento de nuestros sistemas se ven comprometido. Es importante aclarar que cualquier sistema P2P basado en DHT debe proveer un mecanismo óptimo de re-balanceo del sistema para evitar este tipo de problemas independiente si existen nodos maliciosos o no.

SIMULACIÓN DE REDES P2P CON 5000 NODOS

para este trabajo hemos simulado 2 redes p2p, en las cuales el proceso de asignación de los Identificadores de los nodos de la red, es generado de una forma aleatoria por el proceso de asignación de identificadores de cada protocolo de enrutamiento, para esto utilizamos los protocolos P2P llamados Chord y uno más reciente llamado FRT-Chord.

A continuación le mostramos el contenido de los archivos planos que contienen los respectivos comandos utilizados para generar los escenarios antes planteados.

El objetivo de este trabajo no es más que promediar el número de saltos de una red P2P con el proceso de asignación de cada protocolo utilizado en este caso Chord y FRT-Chord, y comparar los resultados con el promedio de número de saltos de una red P2P en la cual el proceso de asignación de los ID no es el adecuado para nuestro caso los ID de los nodos en la red serán consecutivos.

CHORD

```
class ow.tool.msgcounter. Main
#arg -p 10000
schedule 0 invoke timeoffset 1000
# invoke class
 ow.tool.dhtshell.
Main#class ow.tool.groupshell.
Main arg -m emu0 -r Recursive -a Chord -p 8000
schedule 0 invok
arg -m emu0 -r Recursive -a Chord
schedule 5000,1,4999 invoke
# status
#scheduledaemon 0,10000 control 0 status
# join
timeoffset 20000
include C:\Users\MEZA\Desktop\10000-nodes-scenario\nuevojoin2.txt
timeoffset 250000
include C:\Users\MEZA\Desktop\10000-nodes-scenario\put5000.txt
#schedule inf control all halt
```

FRT-Chord

```
class ow.tool.msgcounter.Main
arg -p 10000
schedule 0 invoke
timeoffset 1000
class ow.tool.dhtshell.Main


arg -m emu0 --web -r Recursive -a FRT-Chord -p 8000
schedule 0 invoke
arg -m emu0 -r Recursive -a FRT-Chord -p 8001
schedule 5000 invoke
arg -m emu0 -r Recursive -a FRT-Chord -p 8002
schedule 10000 invoke
arg -m emu0 -r Recursive -a FRT-Chord
schedule 15000,1,4997 invoke
timeoffset 25000
include C:\Users\MEZA\Desktop\10000-nodes-scenario\nuevojoin2.txt
timeoffset 28000
include C:\Users\MEZA\Desktop\10000-nodes-scenario\put2.txt
#schedule inf control all halt
```

Descripción de los escenarios

En estos escenarios se ejecuta un nodo con una aplicación shell del simulador escuchando por el puerto 8000, además se ejecutan otros 4999 de forma automática en un periodo de tiempo especificado.

Los archivos incluidos son el join de la red y el archivo que contiene los mensajes put.

Además que el estilo de enrutamiento a utilizar es recursivo.

Ahora veremos brevemente que contienen dichos archivos mencionadoss anteriormente:

Join

```
schedule 0 controls 2 init emu1
schedule 20 controls 3 init emu1
schedule 40 controls 4 init emu1
schedule 60 controls 5 init emu1
```

Esto es lo que contiene el archivo join como vemos, el **shedule** y el **control** son comandos del simulador, que no se detallan pues no son de interés, sólo observamos los números consecutivos del uno al 10, esto quiere decir que se unirá una red con 10 nodos, para nuestro caso estas líneas se multiplican hasta llegar consecutivamente de 2 a 5000; el em1 no es más que la simulación de la IP del computador.

Este archivo join es contenido en un archivo comprimido, previamente descargado de la página del simulador overlay weaver http://overlayweaver.sourceforge.net/doc/tutorial/emulator/10000-nodes-scenario.tar.gz.

Este escenario es una simulación con 10000 nodos es decir que el join es de 2 a 1000; si queremos simular más nodos pues obviamente tendremos que seguir agregando líneas de comando con los números consecutivamente como hemos visto.

Put

```
schedule 0 control 2708 put k0 v0
schedule 1 control 2708 status
```

Como vemos acá aparecen también el comando schedule y control, estos dos comandos son indispensables para ejecutar la línea o la instrucción, pero no los detallaremos como hemos dicho anteriormente.

después del comando control vemos un número entero y como vemos están desorganizados, esto tiene una razón esos números son los identificadores de los nodos de la red es decir los **emu** de cada nodo es decir, el 2708 es el nodo **emu:2708**, y nos preguntamos ¿cuál es el objetivo de los identificadores de red aleatoriamente?, la respuesta es sencilla así obtendremos valores estadísticos más precisos de la red pues así se indica que se pueden hacer un mensaje put desde cualquier parte de la red, así nos daremos cuenta finalmente que el

promedio del número de mensajes a usar es el mismo; los **put** son mensajes que le dicen a un nodo que guarde un valor V en el nodo responsable de K, pues como sabemos las responsabilidades de los nodos es una característica especial de las redes P2P; donde v es cualquier cosa desde un documento, hasta un archivo mp3.

Luego en la siguientes líneas vemos el mismo número del emu y un comando **status**, con status vemos en detalle información del nodo que envía el mensaje como la tabla de enrutamiento, además de los datos que utilizamos para obtener las estadísticas de la red como el número de mensajes generados por el put y la longitud de los saltos utilizados para enviar dichos mensajes.

Para realizar el archivo put que se incluye en cada escenario, tuvimos la necesidad de generar un script en java que genera los emu de forma aleatoria mediante una función y se llenara un archivo con los datos que lleva el mensaje put.

Lectura de los Logs de la ejecución y abstracción de datos

Los logs son archivos que llevan el resultado que arroja el simulador ante nuestro escenario, es decir son el resultado del conjunto de instrucciones que ponemos a ejecutar en él; estos resultados, son arrojados por el simulador en el output del IDE Netbeans y luego guardados en un archivo con su ruta especificada.

Luego este archivo es leído por un script realizado por nosotros, que hace o toma datos del log los almacena en variables numéricas para sacar el promedio de saltos de la red.

El contenido de un archivo LOG es el siguiente:

Last keys & routes:
number of messages: 15 -> 15
key[0]: 54ad33600d3f80fc155e9ac4664106f34d04ef0e
route[0] (length: 7): []

Los datos que nos interesan a nosotros es contar el número de saltos que da la red para enviar los mensajes esto lo vemos en:

Last keys & routes:
number of messages: 15 -> 15
key[0]: 54ad33600d3f80fc155e9ac4664106f34d04ef0e
route[0] (length: 7):

Lo que hace nuestro script es sacar el número que contiene el **length**, que en este caso es 7 y sumarlo en un contador, al igual que un contador que sume cada vez que en nuestro log recorrido línea por línea comience por **route[0]**, como son 5000 nodos, entonces son 5000 put al igual que los mismos números de status, por ende son 5000 veces que aparece **length.**

Luego se promedia el contador de los números obtenidos por en el **length** entre el contador del número de veces que nuestra línea recorrida comience por **route[0].**

Análisis del escenario con 5000 nodos con ID consecutivos

#class ow.tool.visualizer.Main
class ow.tool.msgcounter.Main
#arg -p 10000
schedule 0 invoke
timeoffset 1000
class ow.tool.dhtshell.Main
timeoffset 2000
include C:\Users\MEZA\Desktop\10000-nodes-scenario\generado.txt
timeoffset 17000
include C:\Users\MEZA\Desktop\10000-nodes-scenario\join1000.txt
timeoffset 200000
include C:\Users\MEZA\Desktop\10000-nodes-scenario\put1000.txt
#schedule inf control all halt

Como podemos ver este es el escenario que se ejecutara para que los ID, sean asignados de forma consecutiva. Este escenario se diferencia en muy poco de los dos estudiados anteriormente; la diferencia consiste en que en este escenario se incluye un nuevo archivo que es el que creara y asiganara los ID de los nodos de la red; esta inclusión se hace con el comando include, el cual llama un archivo en una ruta especifica.
A continuación veremos el contenido del nuevo archivo incluido en el escenario:

arg -m emu0 -r Recursive -a Chord -i 00000000000000000000000000000000000000001

schedule 11 invoke

Este es el contenido de archivo que no es más que la creación manual de un nodo como lo hicimos en los 2 escenarios anteriores, con la diferencia en que en los escenarios pasados se le daba un parámetro –p seguido

de un, esto me indica que el nodo tiene una aplicación Shell del simulador escuchando por ese número de puerto asignado. Ahora nuestro archivo contiene un parámetro –i, este parámetro le indica que ese nodo va a tener como ID en la red ese número que le antecede al parámetro si es **uno** seria **0000000000000000000000000000000000000001**, este número está escrito en hexadecimal.
Para generar un archivo tan extenso manualmente, es demasiado tedioso, por ende se realizó un script en java para generar este archivo de forma automática, y requiere como parámetros la ruta donde se va a guardar el archivo y el número de nodos de la red.

Resultados
Luego de realizar todo el proceso definido anteriormente se leyeron los logs de las ejecuciones de cada escenario. Nuestro script arrojo las siguientes estadísticas.

El escenario de **Chord** arrojo un promedio de saltos de la red con 5000 nodos de 8, mientras el protocolo **FRT-Chord** arrojo un numero de saltos de 7, pues como vemos las diferencias no son muchas y **FRT-Chord** puede ser más eficiente en cuanto al número de saltos de la red se refiere porque es un protocolo que es relativamente nuevo von respecto a Chord, ya que Chord es uno de los primeros protocolos de sistemas basados en **DHT**.
Estas diferencias suelen ser cosas técnicas de cada protocolo y que en este trabajo no compete su análisis.

En cuanto al **escenario con 5000 nodos con ID consecutivos**;
En este escenario los resultados fueron los esperados, pues cuando se precedía a enviar los mensajes de unión a ciertos nodos, estos paquetes de unión o **Join** al igual que los de confirmación de unión **ACK-Join**, no eran enviados o confirmados por el destinatario, simplemente porque el tiempo de vida de dichos paquetes en la red (TTL) se agotaba o expiraban pues el número de saltos de la red era demasiado grande.

SOLUCIONES A ATAQUES DE IDENTIDAD
Para dar solución a estos tipos de ataques, se debe tener en cuenta características en cuanto al proceso de identificación de un nodo en la red se refiere.
La primera característica que se debe estudiar es de asignación, ***la asignación*** estudia como un nodo obtiene un identificador en la red, la segunda es la ***verificación***, esta característica es basada en identificar o verificar la veracidad de los identificadores en la red y la tercera y última es ***la de limitación*** con esta última característica se mitigan ataques como el Sybil, pues se plantea la restricción de identificadores por participante en la red.
Si se quiere dar una solución a los ataques de identidades en las redes P2P, se debe pensar una solución para los problemas que se puedan presentar en las características planteadas anteriormente.
Existen muchos tipos de opiniones para tratar de mitigar este tipo de ataques, pero según Douceur [15], la mejor forma de controlar o mitigar ataques como el de asignación arbitraria de identificadores y Sybil, es mediante el uso de entidades centrales confiables, que asuman el rol de controlar las características de asignación, verificación y limitación que lleva un proceso honesto de identificación.

La estrategia de Castro [16], es basada en el estudio hecho por Douceur [15], pues Castro delega el proceso de asignación de identificadores a una entidad certificadora confiable, dicha entidad genera un certificado que contiene: un identificador generado de forma aleatoria, una llave pública y la dirección IP del nodo. Todos los nodos de la red tienen conocen la llave publica de la entidad certificadora, esto se hace con el objeto de preguntarle a la entidad si un certificado es o no valido. Los nodos con un certificado valido pueden ingresar y colaborar en el proceso de enrutamiento, además pueden salir de la red si preguntarle a la entidad certificadora.

El proceso de limitación de los ID, importante para mitigar el ataque Sybil, es una propuesta interesante, Castro propone que se los certificados generados tengan un valor económico, pues si algún atacante quiere tomar posesión de la red, le costara un dinero proporcional a la cantidad de nodos que nuestra red posea. El autor también propone como una opción ligar los identificadores a entidades reales como la cedula del participante en la red, esto garantizaría la no repetición de los identificadores.
Pues nuestro grupo ha llegado a la conclusión que la estrategia planteada por castro es sin duda interesante, pero se puede mejorar, además de todas las características de la estrategia antes planteada, se puede introducir el uso de métodos como **captcha**, para revisar disponibilidad del nodo, esto se haría en un límite de tiempo, y si el nodo no resuelve el acertijo será sacado de inmediato de la red.
Esto es con el objetivo de mitigar el ataque Sybil pues el atacante tendría que ir computador a computador para resolver el acertijo y si la cobertura de la red por parte del atacante es demasiada, algunos de sus nodos no alcanzara el proceso de verificación de disponibilidad y

será excluido de la red; con estos métodos el atacante disminuiría su cobertura en la red.
Existen otros tipos de estrategias como las estrategias jerárquicas planteadas en [17] y las estrategias descentralizadas detalladas en [18].

CONCLUSIONES

Analizando los resultados arrojados por el simulador OVERLAY WEAVER podemos denotar que la asignación de identificadores de cada uno de los nodos conectados a una red P2P especifica es un proceso que debe ser realizado de forma segura, debido que gracias a esto depende el buen funcionamiento de dicha red; si no se acatan los procesos de asignación puede influir notoriamente en la perdida de paquetes enviados a través de la red. Por todo lo mencionado anteriormente en el desarrollo del artículo nos damos cuenta que al mantener los nodos enlazados de forma secuencial afecta directamente la red P2P, ya que los saltos que realiza cada paquete enviado por un nodo específico suelen ser demasiados extensos. Lo cual ocasiona que el tiempo de vida de cada paquete enviado se agote, lo que significa que el nodo destino no reciba dicha información. Ésta situación no es más que una muestra especifica de un ataque de DENEGACION DE SERVICIOS DISTRIBUIDOS (DDOS).

Agradecimientos

Los autores reconocen la contribución del docente R. Villanueva por su constante apoyo y por habernos guiado en este proceso de aprendizaje

REFERENCIAS

[1] D. Stinson, Cryptography Theory and Practice (Discrete Mathematics and its Applications). Chapman & Hall/CRC. 2006.

[2] S. Ratnasamy, P. Francis, M. Handley, R. M. Karp, S. Shenker: A scalable content–addressable network. In: SIGCOMM. 2001.

[3] S. Androutsellis-Theotokis, D. Spinellis, A survey of peer-to-peer content distribution technologies. ACM Comput. Surv. **36**(4), 335–371. 2004.

[4] I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M.F. Kaashoek, F. Dabek, H. Balakrishnan, Chord: a scalable peer-to-peer lookup protocol for internet applications. IEEE/ACM Trans. Netw. 11(1), 17–32. 2003.

[5] http://www.freepastry.org/PAST/pastry.pdf

[6] P. Druschel and A. Rowstron. PAST: A large-scale, persistent peer-to-peer storage utility. In *Proc. HotOS VIII*, Schloss Elmau, Germany, May 2001.

[7] A. Rowstron and P. Druschel. Storage management and caching in PAST, a large-scale, persistent peer-to-peer storage utility. In *Proc. ACM SOSP'01*, Banff, Canada, Oct. 2001.

[8] A. Rowstron, A.-M. Kermarrec, P. Druschel, and M. Castro. Scribe: The design of a large-scale event notification infrastructure. Submitted for publication. June 2001. http://www.research.microsoft.com/antr/SCRIBE/.

[9] P. Maymounkov, Mazieres D. Kademlia: A peer-to-peer information system based on the XOR metric. In Proc of IPTPS02, Cambridge, USA, March 2002.

[10] T. Holz, M. Steiner, F. Dahl, E. W. Biersack, and F. Freilin Measurements and mitigation of peer-to-peer-based botnets: A case study on storm worm. In LEET, 2008.

[11] K. Needels, M. Kwon: Secure routing in peer-to-peer distributed hash tables. In:SAC'09,ACM, pp.54-58.

[12]P. Wang, I. Osipkov, N. Hopper, Y. Kim: Myrmic: secure and robust DHT routing, Submission, 2007.

[13] X. Xiang, T. Jin: E_cient secure message routing for structured peer-to-peer systems.In:NSWCTC'09, IEEE, pp.354-357.

[14] Seguridad en Sistemas P2P dht, Ricardo Luis Villanueva Polanco, Universidad de los Andes, Facultad de Ingenieria; Tesis de grado 2010.

[15] J. Douceur: The sybil attack. In:IPTPS'02, Springer, pp.251-260.

[16] M. Castro, P. Druschel, A. Ganesh, A. Rowstron, D. Wallach: Secure routing for structured peer-to-peer overlay networks. In:SIGOPS'02, ACM, pp.299-314.

[17] H. Rowaihy, W. Enck, P. Mcdaniel , y T. La Porta,. Limiting sybil attacks in structured P2P networks. In Proc. of 26th IEEE Int'l Conference on Computer Communications. IEEE. 2007

[18] I. Baumgart, S. Mies, S. Kademlia: A practicable approach towards secure key-based routing. Proceedings of the 13th Int'l Conf. on Parallel and Distributed Systems, IEEE, pp.1-8. 2007