GESTIÓN DE LA REPUTACIÓN EN REDES P2P: ALGORITMO DE REPUTACIÓN DE P2PEOPLE

Rafael Melendreras Ruiz, Rafael Berenguer Vidal, Ángel J. García Collado
Departamento de Ciencias Politécnicas – Grupo de Inv. en Telecomunicaciones Avanzadas
Universidad Católica San Antonio de Murcia
e-mail: [rmelendreras, rberenguer, ajgarcia]@pdi.ucam.edu

***Abstract*- In this paper, a new reputation management system is presented for a peer-to-peer platform called P2People. Due to peer-to-peer applications providing anonymity features, malicious peers collectives try to subvert these systems under self interests. So, additional mechanisms need to be implemented to increase the security at each node of the network. Based on human perception of trustworthiness, the P2People reputation algorithm tries to get accurate results in order to minimize the number of downloads and transactions from/with malicious peers, and at the same time to isolate them. The simulation results shown that this first approach gets to reduce hugely the number of corrupted files and the contact with undesirable peers in the network.**

I. INTRODUCCIÓN

Las redes P2P presentan un esquema de relación simétrica entre sus nodos (*peers*). Cada máquina conectada es dotada de una mayor autonomía, pasando a comportarse tanto como cliente como servidor, lo cual obliga a cada usuario del sistema a proveer acceso a parte de sus recursos [1].

Existe una visión sesgada de la red porque todos los usuarios activos en la misma difícilmente están interconectados entre sí. La red posee una naturaleza probabilística, debido a que los nodos presentan esquemas de conectividad variables. A su vez, existe un alto grado de accesibilidad a los recursos de la red.

La descentralización de la red elimina la vulnerabilidad que en otras arquitecturas se concentraba exclusivamente en un único punto (el servidor centralizado), ya que existen virtualmente tantos servidores como usuarios conectados en ese momento a la misma.

Al mismo tiempo, se precisa un volumen mayor de comunicaciones entre los nodos, lo cual repercute notablemente en la carga de tráfico de la red.

Por otro lado, cada vez es más frecuente en aplicaciones de tipo P2P preservar el anonimato de los *peers*, lo cual fomenta la aparición de comportamientos indeseables por parte de ciertos usuarios con el fin de degradar el funcionamiento de la red. Los sistemas de gestión de la reputación se destinan a favorecer la detección y el aislamiento de este tipo de usuarios.

II. ESTADO DEL ARTE

A. Aplicaciones P2P

Actualmente, existen diversas implementaciones de sistemas *peer-to-peer* en el mercado. Tales soluciones suelen centrarse en tareas específicas. Por ejemplo, Gnutella [2] se destina a la transferencia de ficheros de manera distribuida. Napster [3], se utiliza para el intercambio de ficheros de música. Los productos basados en el protocolo fasttrack (Morpheus, Kazaa [4], etc) están especializados en la distribución de contenidos multimedia.

Existen otros proyectos centrados en aplicaciones distintas como la comunicación de voz y datos, o el comercio B2C como es el caso de 1stWorks. También pueden encontrarse soluciones cuyo fin es la gestión de bases de datos distribuidas, como Mariposa [5]. A diferencia de los anteriores, el proyecto P2People trabaja con la formación de "grupos de intereses comunes" para posteriormente brindar a sus usuarios de servicios colaborativos P2P [6].

B. Principios de la reputación P2P

La mayoría de aplicaciones que proveen acceso a redes *peer-to-peer* usan técnicas o métodos de ocultación de IP, para proteger así la identidad de sus usuarios. Es lo que se conoce como anonimato (*anonimity*) [7].

Dicho factor propicia que tales redes tiendan a ser atacadas por usuarios malintencionados (*malicious peers*), con el objetivo de introducir archivos infectados con virus o archivos manipulados (*inauthentic files*). Éstos últimos no se corresponden con el contenido esperado, están incompletos o no funcionan. La estrategia de distribución de tales archivos persigue decrementar el uso de la red en base a mermar la confianza de los usuarios.

La mayoría de los nodos de las redes P2P, son máquinas de muy bajas prestaciones en comparación con los servidores de arquitecturas cliente-servidor. Su capacidad para afrontar los ataques procedentes de usuarios malintencionados es pequeña, lo cual se ve agravado por la autonomía que les confiere la red y su carácter distribuido. Tal situación transfiere a las aplicaciones P2P gran parte de la responsabilidad en cuanto a la prevención contra dichos ataques. La solución pasa por el desarrollo de sistemas de gestión de la reputación, cuyo objetivo es minimizar el número de contactos -sesiones de trabajo o colaboración- con usuarios malintencionados, en base a la detección y el

aislamiento de los mismos. A tal efecto, se desarrollan algoritmos que permiten a cada usuario estimar la fiabilidad de otros, en función de su opinión y la de sus conocidos. La opinión que un *peer* tiene sobre otro equivale al valor de su último voto tras haber interaccionado con éste -transaccionado, compartido archivos, etc-.

C. Diseño de sistemas de gestión de la reputación P2P

A la hora de diseñar un sistema de gestión de la reputación *peer-to-peer*, las consideraciones más importantes son [8]:

1. El sistema debe ser autogobernado. Los propios *peers* deben ser quienes definan y se encarguen de hacer respetar las reglas de funcionamiento del sistema, y no una autoridad central.
2. El sistema debe mantener el anonimato. La reputación de un *peer* debe asociarse con un identificador opaco más que con identidades externamente asociadas (tales como la dirección IP de un usuario).
3. El sistema no debe beneficiar a los nuevos usuarios. La reputación debe conseguirse mediante un buen comportamiento a lo largo de la vida del *peer* en la red -conjunto de interacciones positivas-. No debería ser una ventaja para los *peers* maliciosos (con pobres reputaciones) el estar cambiando constantemente sus identidades en la red para conseguir el estatus de nuevo usuario.
4. El sistema debe disponer de una capacidad mínima en términos de procesado, infraestructura, ancho de banda de almacenamiento y complejidad de mensajes.
5. El sistema debe ser robusto frente a colectivos de *peers* maliciosos quienes se conocen y tratan de degradar al sistema de manera conjunta.
6. El ciclo de vida de la reputación, es el periodo en el que los mensajes recibidos deben permanecer almacenados para ser considerados por el sistema de reputación.
7. El problema de "arranque en frío" puede ser resuelto si los nuevos *peers* pueden participar en la distribución de recursos bien conocidos.
8. Antes de empezar a desarrollar, los cuellos de botella del funcionamiento de la aplicación deben considerarse en el algoritmo.
9. En cuanto un recurso corrompido sea detectado, debe ser asociado al *peer* que lo posea. Esto consiste en crear una estrategia de "listas negras".

D. Sistemas de gestión de la reputación actuales

Muchas aplicaciones desarrollan, en función de los servicios a los que están enfocadas, distintos algoritmos para el cálculo de la reputación. Algunos de los más significativos son:

- **eBay**

El sistema de reputación de eBay es realimentado, ya que tanto los compradores como los vendedores de este sitio de subastas (*auction site*) pueden votarse entre sí después de cada transacción, positivamente ($tr(i,j)=1$) o negativamente ($tr(i,j)=-1$). Si se reciben varias votaciones de un mismo usuario, sólo será tenida en cuenta la última de ellas. La reputación global de cada participante consiste en el número de votos positivos recibidos de usuarios distintos dentro de los últimos seis meses. P.e. Tomás (24), significa que al usuario cuyo id es Tomás le han votado 24 personas distintas positivamente. Un sistema centralizado almacena y computa dichas votaciones para extraer los resultados [9].

- **Kazaa**

Kazaa es una aplicación de intercambio de ficheros P2P (*file sharing*) [4]. Su sistema de gestión de la reputación se centra en los recursos y no en los usuarios. Existe una primera fase denominada "Evaluación de Integridad" (*Integrity Rating*), no obligatoria, donde se permite al usuario evaluar a cada uno de sus archivos compartidos mediante 4 niveles -excelente, medio, pobre, borrar archivo-. En la segunda fase se calcula el "Nivel de Participación" (*Participation Level*), que indica el grado de uso de la aplicación por parte del usuario. La expresión para su cálculo es la siguiente:

$$plevel_i = \frac{uploaded_i}{downloaded_i} x100 \quad (1)$$

Se tiene que $uploaded_i$ es la información descargada del usuario i en Mbytes por otros usuarios. Las descargas de archivos que hayan sido evaluados en la fase 1 por i se contabilizan como del doble del valor de su capacidad en Mbytes. Por otro lado, $downloaded_i$ es la información descargada por i de otros *peers* en MBytes.

El valor de $plevel_i$ se asocia a una categoría de entre 6 posibles. Cada categoría representa un número de fuentes de descarga adicionales que la aplicación presenta al usuario como recompensa. Cuanto mayor sea el nivel de participación, mayor es la categoría a la que se pertenece y en consecuencia, mayor es el número de fuentes de descarga disponibles en la aplicación.

- **EigenTrust**

En el sistema *Eigen Trust* [8], se define el valor de confianza local s_{ij} como la suma de todas las transacciones individuales que el *peer i* ha descargado del *peer j*: $s_{ij}=\Sigma tr_{ij}$. Por tanto, cada *peer i* puede almacenar el número de transacciones satisfactorias que ha tenido con el *peer j*, *sat(i,j)* y el número de transacciones insatisfactorias que ha tenido con este, *unsat(i,j)*. Luego, s_{ij} se define:

$$s_{ij} = sat(i,j) - unsat(i,j) \quad (2)$$

Para evitar valores exagerados de confianza local, se normaliza dicho parámetro a un valor c_{ij}:

$$c_{ij} = \frac{\max(s_{ij},0)}{\sum_j \max(s_{ij},0)} \quad (3)$$

A diferencia de los anteriores, este sistema de reputación recoge los valores de confianza local de todos los usuarios de una forma natural y con el mínimo coste en términos de complejidad de mensaje. Esta aproximación se basa en la noción de confianza transitiva: un *peer i* tendrá una buena opinión de aquellos *peers* que le hayan proporcionado archivos no corrompidos. Además, dicho *peer* tenderá a confiar en las opiniones de esos mismos usuarios, debido a que si han sido honestos con los archivos que ellos mismos proveen deberán serlo en cuanto a la información que reporten de sus valores de confianza local. El valor de reputación global que un *peer i* asigna a otro *k* en función de la opinión de sus conocidos se expresa como:

$$t_{ik} = \sum_j c_{ij} c_{jk} \quad (4)$$

dónde c_{ij} expresa el valor de confianza local del *peer i* en su conocido *j*, y c_{jk} la opinión de su conocido *j* en el *peer k*.

En notación matricial, y tras tener en cuenta hasta *n* niveles de conocidos se tiene que:

$$\vec{t}_i = ([c_{ij}]^{\mathrm{T}})^n \vec{c}_i = (C^T)^n \vec{c}_i \quad (5)$$

Notar que los valores de reputación t_{ik}, t_{jk} que dos usuarios distintos *i* y *j* calculan para un mismo usuario *k* no tienen por qué coincidir, debido a que tanto sus conocidos como las opiniones de éstos serán distintos.

III. ALGORITMO DE REPUTACIÓN DE LA APLICACIÓN P2PEOPLE

A. Principios de funcionamiento

A diferencia de la gran mayoría de aplicaciones P2P, la plataforma P2People ofrece a sus usuarios una amplia gama de servicios colaborativos, entre los que destacan el comercio electrónico C2C (*e-commerce*) o la compartición de escritorio (*desktop sharing*) [6]. Dicha circunstancia, unida al uso del anonimato por parte de la aplicación, justifica el desarrollo de un sistema propio de gestión de la reputación. El algoritmo de reputación diseñado hereda el concepto de "Confianza Transitiva" del algoritmo Eigen Trust, aunque como veremos en su código sólo se implementa hasta un primer nivel de conocidos n=1. A diferencia de aquél, se tienen en cuenta también los valores de confianza local de usuarios desconocidos. Asimismo, se introduce una limitación según la cual sólo se permite votar una sola vez al día a un mismo usuario y, además, el sistema de votaciones permite introducir valores de reputación que abarcan desde 0 (peor) hasta 4 (excelente). Cada *peer* posee un vector en el que almacena todos y cada uno de los votos que ha emitido al resto de *peers*. En su estructura, se almacenan los votos realizados a cada *peer*, acompañados de su identificador de usuario correspondiente.

Votos emitidos $\vec{v}_{ve}$ *:[(id_rxtor₁,voto₁), (id_rxtor₂, voto₂), ...]*

También se almacena un vector de votos recibidos, con una estructura similar al de votos emitidos. La información de este vector sólo es accesible por el resto de *peers* del sistema, para evitar posibles manipulaciones por parte de *peers* malintencionados con la intención de incrementar su valor de reputación. Su forma es:

Votos recibidos $\vec{v}_{vr}$ *:[(id_trxor₁,voto₁), (id_trxor₂, voto₂), ...]*

Cada usuario comparte parte de su información de reputación (vectores de votos recibidos) con otros en base a un sencillo protocolo de encuesta distribuido (*distributed polling*) [10]. Un usuario, obviamente, no puede votarse a sí mismo.

El coste computacional del algoritmo radica en la adquisición distribuida y el procesado de los vectores de información (votos emitidos y recibidos) que almacena cada *peer*. Dicho coste se reduce gracias a la agrupación de los usuarios en áreas temáticas y al filtrado de *matching* realizado por la aplicación, que sólo permite penetrar en nuestra aplicación a aquellos usuarios con un perfil de intereses cuyo porcentaje de afinidad con el nuestro supere un umbral preestablecido. El algoritmo sólo se aplica a los usuarios presentados por la aplicación en la GUI principal.

La formulación empleada para el cálculo de la reputación de un usuario de la red -al que denominaremos *peer0*-, se apoya fundamentalmente en tres parámetros, cuyo valor por defecto es -1:

1. El valor de nuestra opinión sobre el *peer0*. Será tenida en cuenta nuestra última votación en el caso de existir alguna. Almacenada en la variable *repA*:

$$repA = stack[length() - 1] \quad (6)$$

La instrucción *stack(length()-1)* hace referencia al último valor contenido en la pila de mis votos emitidos al *peer0*.

2. El valor que representa para nosotros la opinión sobre el *peer0* de aquellas personas con las que hayamos interaccionado alguna vez (conocidos). Se almacena en la variable *repB*:

$$repB = \frac{\sum_0^{length(v_{1,2})-1} \left(\frac{v_1[i]}{4} \cdot v_2[i] \right)}{length(v_{1,2})} \quad (7)$$

donde v_1 es el vector en el que se almacena nuestra última votación a cada uno de nuestros conocidos que han votado alguna vez al *peer0*. Se obtiene extrayendo de nuestro vector $\vec{v}_{ve}$ la última votación efectuada a cada uno de los usuarios contenidos en él. v_2 es el vector en que se almacena la última votación de cada uno de nuestros conocidos al *peer0*. Se obtiene extrayendo del vector $\vec{v}_{vr}$, la última votación recibida de cada uno de los usuarios cuyo id se encuentre contenido. Como consecuencia, la longitud de los vectores v_1 y v_2 han de ser las mismas.

3. El valor de la opinión sobre el *peer0* de aquellas personas con las que nunca hayamos interaccionado (desconocidos). Se almacena en la variable *repC* y contiene la media de las últimas votaciones efectuadas por dichas personas al *peer0*. En v_3 se almacenan dichas últimas votaciones, extraídas del vector $\vec{v}_{vr}$:

$$repC = \frac{\sum_0^{length(v_3)-1} (v_3[i])}{length(v_3)} \quad (8)$$

Notar, que previamente al proceso de cálculo, se ha debido llevar a cabo una petición al *peer0* de su vector de votos recibidos, lo cual lleva asociado un coste temporal y a la vez un consumo de recursos de red en forma de mensajes de petición y respuesta.

La reputación final sobre un usuario se obtiene como función de estos tres parámetros ponderados por unos coeficientes α_A, α_B y α_C:

$$rep = f(\alpha_A repA, \alpha_B repB, \alpha_C repC) \quad (9)$$

En el diseño del algoritmo, se ha considerado *f* como la función suma, debido a que *rep* se obtiene añadiendo las aportaciones con las que cada tipo de *peer* -uno mismo, conocidos y desconocidos- contribuye para forjar nuestra opinión final sobre un usuario. Los coeficientes significan el grado de confianza depositado en la opinión de cada tipo de *peer*. Son optimizados mediante simulaciones controladas con la herramienta Matlab, basadas en el estudio de la evolución de poblaciones de usuarios con distintos patrones de reputación y con distintos rangos de actividad, agrupados en tipos de comportamiento.

El rango de valores de la reputación de un usuario *rep*, es idéntico al de las votaciones, es decir, abarca desde 0 hasta 4.

Cuando un usuario se estrena en la aplicación no lo hace con la reputación más baja, sino con un valor igual a 1, para solucionar el típico problema de "arranque en frío", y controlar la actividad de los *freeriders* [11].

B. Resultados del algoritmo

Los siguientes coeficientes, son los que se han implementado en el código del algoritmo de reputación de la aplicación P2People. Han ofrecido los mejores resultados en cuanto a la detección y discriminación de usuarios y agrupaciones de éstos malintencionados. Son los siguientes:

repA	repB	repC	α_A	α_B	α_C
≠-1	≠-1	≠-1	0,65	0,3	0,05
≠-1	≠-1	=-1	0,7	0,3	0
≠-1	=-1	≠-1	0,85	0	0,15
≠-1	=-1	=-1	1	0	0
=-1	≠-1	≠-1	0	0,7	0,3
=-1	≠-1	=-1	0	1	0
=-1	=-1	≠-1	0	0	1
=-1	=-1	=-1	0	0	0

Tabla 1. Fórmulas de Reputación -rep- para los distintos casos del algoritmo.

En la figura 1 se observa, que el algoritmo funciona bien para agrupaciones de hasta un 45% inicial de usuarios malintencionados, tras dos semanas de funcionamiento de la aplicación y sobre una población de 10000 usuarios con una actividad media de 4 contactos al día.

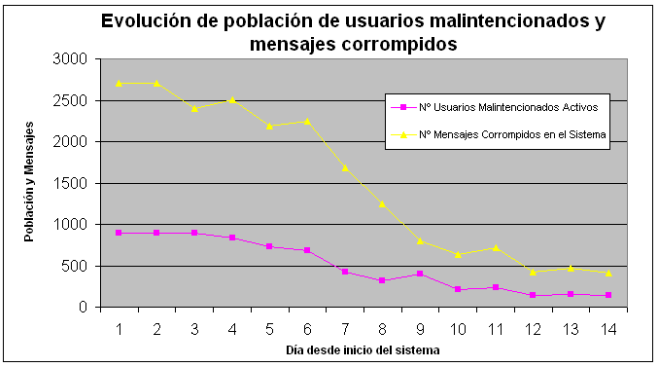

Fig. 1. Estudio de la evolución de usuarios malintencionados y mensajes para una población de 10000 usuarios y un porcentaje inicial de éstos del 45%.

Otro importante resultado son las prestaciones del algoritmo en cuanto al consumo de recursos de red. Algunas aplicaciones P2P incorporan sistemas de adquisición de datos de manera distribuida para calcular la reputación, que consisten en ir preguntando *peer* a *peer* su opinión acerca del *peer0* y esperar a recibir el mayor número de respuestas posibles.

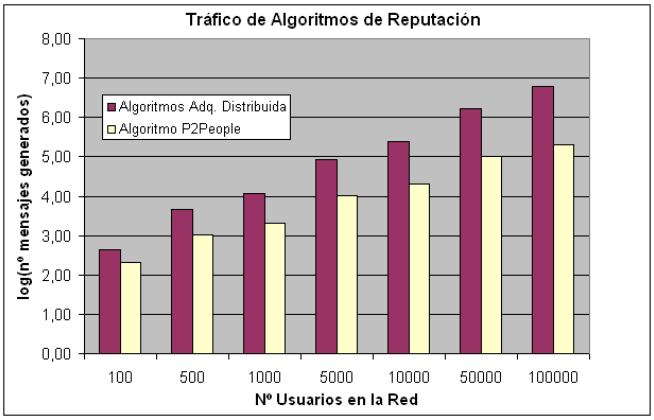

Fig. 2. Comparación del tráfico generado por distintos algoritmos de reputación en función del número de usuarios de la red.

Sin embargo, cada vez que un usuario ejecuta el algoritmo de reputación de P2People, solo precisa generar un único tráfico en dirección al *peer0* con la intención de obtener su vector de votos recibidos $\vec{v}_{vr}$. En la figura 2, se muestra la diferencia entre los tráficos de cada tipo de algoritmo. La única contrapartida, es la necesidad de actualización y mantenimiento los vectores de votos, tanto $\vec{v}_{vr}$ como $\vec{v}_{ve}$.

IV. CONCLUSIONES

En este artículo ha sido justificada la necesidad de los sistemas de gestión de la reputación para las aplicaciones basadas en redes P2P que incorporen técnicas de ocultación de IP o anonimato.

El algoritmo P2People, se centra en la reputación de personas y no en la de los recursos. La idea original de su diseño parte del concepto de "Confianza Transitiva" del algoritmo Eigen Trust, en el que la opinión de personas conocidas ayuda a obtener una descripción más exacta de cómo son otros usuarios. La opinión de usuarios desconocidos son tenidas en cuenta también.

La agrupación de usuarios en áreas temáticas y el filtrado de *matching* que incorpora la plataforma P2People, y la creación de los vectores de votos emitidos y recibidos consiguen mejorar sus prestaciones en cuanto al consumo de recursos de red y la eficiencia temporal.

Los resultados de la ejecución del algoritmo de reputación, además de aportar información confiable a los usuarios, alcanzando a detectar y reducir agrupaciones de *peers* malintencionados de hasta un 45% del total de la población tras dos semanas de funcionamiento, se usan para implementar aplicaciones de filtrado y para configurar la activación de determinados servicios colaborativos a través de la componente matriz de servicios. De esta forma, se permite personalizar opciones de seguridad de la plataforma.

AGRADECIMIENTOS

Este trabajo ha sido financiado por la Unión Europea dentro del programa IST a través de P2People-2002-38458.

REFERENCIAS

[1] L. Gong, "*Peer-to-Peer* Networks in Action", *IEEE Internet Computing*, vol. 6 issue: 1, pp. 37–39, Jan/Feb. 2002.

[2] Clip2. "The Gnutella Protocol Specification v0.4 (Document Revision 1.2)", Jun. 2001.

[3] Dr. Scholl, "Napster protocol specification". 7 Apr. 2001.

[4] N. S. Good and A. Krekelberg, "Usability and privacy: a study of Kazaa P2P file-sharing". HP Laboratories, Palo Alto, California, USA..

[5] M. Stonebraker, P. M. Aoki, W. Litwin, A. Pfeffer, A. Sah, J. Sidell, C. Staelin, A. Yu., "Mariposa: A Wide-Area Distributed Database System". *VLDB Journal*, vol. 5(1), pp. 48-62, 1996.

[6] R. Melendreras Ruiz, R. Berenguer Vidal and A.J. García Collado. "Desarrollo de la plataforma de colaboración basada en grupos de "intereses comunes" y redes *peer-to-peer*: P2People", *Revista IEEE América Latina*, vol. 2, issue: 1, Mar. 2004.

[7] R. Dingledine, N. Mathewson, P. Syverson, "Reputation in P2P Anonymity Systems", http;//www.freehaven.net.

[8] S. Kamvar, M. Schlosser, H. García Molina, "The Eigen Trust Algorithm for Reputation Management in P2P Networks", *The International World Wide Web Conference*, 20-24 May. 2003, Budapest, Hungary.

[9] J. Patterson, "A Matter of Trust: Reputation Management in *Peer-to-Peer* Networks", *Computer Science Seminar II Conference*, University of Minessota, Dec. 2003.

[10] M. Gupta, P. Judge, M. Ammar, "A Reputation System for *Peer* to *Peer* Networks", *NOSSDAV'03*, 1-3 Jun. 2003, Monterey, California, USA.

[11] E. Damiani, "A Reputation-based Approach for Choosing Reliable Resources in *Peer-to-Peer* Networks". www.limewire.com.