DISTRIBUTED PATH RESTORATION ALGORITHM FOR ANONYMITY IN P2P FILE SHARING SYSTEMS

Pilar Manzanares-Lopez, Juan Pedro Muñoz-Gea, Josemaria Malgosa-Sanahuja
Juan Carlos Sanchez-Aarnoutse and Joan Garcia-Haro
Department of Information Technologies and Communications
Polytechnic University of Cartagena
Campus Muralla del Mar, 30202, Cartagena, Spain

Keywords: Peer-to-peer networks, anonymity, distributed systems.

Abstract: In this paper, a new mechanism to achieve anonymity in peer-to-peer (P2P) file sharing systems is proposed. As usual, the anonymity is obtained by means of connecting the source and destination peers through a set of intermediate nodes, creating a multiple-hop path. The main paper contribution is a distributed algorithm able to guarantee the anonymity even when a node in a path fails (voluntarily or not). The algorithm takes into account the inherent costs associated with multiple-hop communications and tries to reach a well-balanced solution between the anonymity degree and its associated costs. Some parameters are obtained analytically but the main network performances are evaluated by simulation.

1 INTRODUCTION

Peer-to-peer networks are one of the most popular architectures for file sharing. In some of these scenarios, users are also interested in keeping mutual anonymity; that is, any node in the network should not be able to know who is the exact origin or destination of a message. Traditionally, due to the connectionless nature of IP datagrams, the anonymity is obtained by means of connecting the source and destination peers through a set of intermediate nodes, creating a multiple-hop path between the pairs of peers.

There are various existing anonymous mechanisms with this operation, but the most important are the mechanisms based in Onion Routing (Reed et al., 1998) and the mechanisms based in Crowds (Reiter and Rubin, 1999). The differences between them are the following: In Onion, the initiator can determine an anonymous path in advance to hide some identification information. When a user wants to establish an anonymous communication, he will forward its message to a *Onion proxy*. This *Onion proxy* will randomly select some nodes in the network and will establish the complete route (called *Onion*) between the initiator an the responder. This *Onion*, is a recursively layered data structure that contains the information about the route to be followed over a network. Every node can only decrypt its corresponding layer with its private key, therefore, the first node will decrypt the first layer to see the information about next hop in the route, and this process will continue until the message reaches its destination. Tarzan (Freedman and Morris, 2002), Tor (Dingledine et al., 2004), SSMP (Han et al., 2005) and (Xiao et al., 2003) are implemented based on Onion Routing to provide anonymous services.

On the other hand, in Crowds let middle nodes select the next hop on the path. A user who wants to initiate a communication will first send the message to its node. This node node upon receiving the request will flip a biased coin, to decide whether or not to forward this request to another node. The coin decides about forwarding the request based on probability p. If the probability is to forward, then it will forward to another node and the process continues. If the probability is not to forward, then it will directly forward the request to the final destination. Each node when forwarding to another node records the predecessors information and in this way a path is built, which is used for communication between the sender and the receiver. There are several works (Levine and Shields, 2002), (Mislove et al., 2004), (Lu et al., 2004), based in the mechanism used by Crowds to provide anonymity.

Manzanares-Lopez P., Pedro Muñoz-Gea J., Malgosa-Sanahuja J., Carlos Sanchez-Aarnoutse J. and Garcia-Haro J. (2007).
DISTRIBUTED PATH RESTORATION ALGORITHM FOR ANONYMITY IN P2P FILE SHARING SYSTEMS.
In *Proceedings of the Second International Conference on Software and Data Technologies - PL/DPS/KE/WsMUSE*, pages 88-95
DOI: 10.5220/0001338600880095

The above procedure to reach anonymity has two main drawbacks. The first one is that the peer-to-peer nature of the network is partially eliminated since now, peers are not directly connected but there is a path between them. Therefore, the cost of using multiple-hop path to provide anonymity is an extra bandwidth consumption and a additional terminal (node) overhead.

In addition, as it is known, the peer-to-peer elements are prone to unpredictable disconnections. Although this fact always affects negatively the system performances, in an anonymous network it is a disaster since the connection between a pair of peers probably would fail although both peers are running. Therefore, a mechanism to restore a path when an unpredictable disconnection arises is needed but it also adds an extra network overhead in terms of control traffic.

(Wright et al., 2002) presented a comparative analysis about the anonymity and overhead of several anonymous communication systems. This work presents several results that show the inability of protocols to maintain high degrees of anonymity with low overhead in the face of persistent attackers. In (Sui et al., 2003), authors calculate the number of appearances that a host makes on all paths in a network that uses multiple-hop paths to provide anonymity. This study demonstrates that participant overhead is determined by number of paths, probability distribution of the length of path, and number of hosts in the anonymous communication system.

In this paper we propose an anonymity mechanism for a hybrid P2P network presented in a previous work (Muñoz-Gea et al., 2006) based on Crowds to create the multiple-hops path. However, our mechanism introduce a maximum length limit in the path creation algorithm used by Crowds, in order to retrict the partipant overhead.

The main paper contribution is a distributed algorithm to restore a path when a node fails (voluntarily or not). The algorithm takes into account the three costs outlined above in order to obtain an equilibrated solution between the anonymity degree and its associated costs. The parameters are obtained analytically and by simulation.

The remainder of the paper is as follows: Section 2 summarizes the main characteristics of our hybrid P2P architecture. Section 3 and 4 deeply describes the mechanism to provide anonymity. Section 5 shows the simulation results and finally, Section 6 concludes the paper.

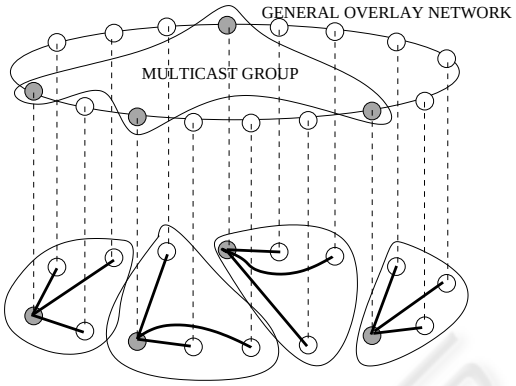

Figure 1: General architecture of the system.

2 A HYBRID P2P ARCHITECTURE

In our proposal (Muñoz-Gea et al., 2006), peers are grouped into subgroups in an unstructured way. However, this topology is maintained by means of an structured lookup mechanism.

Every subgroup is managed by one of the subgroup members, that we call “subgroup leader node”. The subgroup leader is the node that has the best features in terms of CPU, bandwidth and reliability. When searching for a content, the user will send the lookup parameters to its local subgroup leader. The local subgroup leader will resolve the complex query, obtaining the results of this query.

To allow the subgroup leaders to locate contents placed in remote subgroups, all the leaders of the network are going to be members of a multicast group. The structured P2P network can be used to implement an Application Level Multicast (ALM) service. CAN-multicast (Ratnasamy et al., 2001b), Chord-multicast (Ghodsi et al., 2003) and Scribe (Castro et al., 2002) are ALM solutions based on the structured P2P networks CAN (Ratnasamy et al., 2001a), Chord (Stoica et al., 2003) and Pastry (Rowstron and Druschel, 2001), respectively. Anyone of these methods provides an efficient mechanism to send messages to all the members of a multicast group.

To implement the structured-based maintenance of the unstructured topology, all the nodes (peers) must be immersed into a structured network. Therefore, every node has an identifier (*NodeID*), and they have to contact with an existing node to join this network. Figure 1 describes the general architecture of the system.

The previous existing node also gives to the new node the identifier of the subgroup it belongs (*SubgroupID*), and using the structured lookup mechanism

the new node will find the leader of that subgroup. This is possible because each time a node becomes a subgroup leader, it must contact with the node which *NodeID* fits with the *SubgroupID* and sends it its own IP address.

Initially, the new node will try to connect the same subgroup than the existing node. To do that, the new node must contact with the leader of this subgroup, that will accept its union if the subgroup is not full. However, if there is no room, the new node will be asked to create a new (randomly generated) subgroup or it will be asked to join the subgroup that the resquested leader urged to create previously. The use of different existing nodes allows the system to fill up incomplete subgroups. In addition, to avoid the existence of very small subgroups, the nodes will be requested by their leader to join another subgroup if the subgroup size is less than a threshold value.

When a new node finds its subgroup leader, it notifies its resources of bandwidth and CPU. Thus, the leader forms an ordered list of future leader candidates: the longer a node remains connected (and the better resources it has), the better candidate it becomes. This list is transmitted to all the members of the subgroup.

3 PROVIDING ANONYMITY

In this section we propose a file sharing P2P application built over the previous network architecture which provides mutual anonymity. As it is defined in (Pfitzmann and Hansen, 2001), a P2P system provides mutual anonymity when any node in the network, neither the requester node nor any participant node, should not be able to know with complete certainty who is the exact origin and destination of a message.

Our solution defines three different stages. On one hand, all peers publish their contents within their local subgroups (publishing). To maintain the anonymity during this phase, a random walk proceduce will be used. On the other hand, when a peer initiates a complex query, a searching phase will be carried out firstly (searching). Finally, once the requester peer knows all the matching results, the downloading phase will allow the download of the selected contents (downloading). In the following section these three stages are described in detail.

3.1 Publishing the Contents

To maintain the anonymity, our solution defines a routing table at each peer and makes use of a random walk (RW) technique to establish a random path from the content source to the subgroup leader. When a node wants to publish its contents, first of all it must choose randomly a connection identifier (great enough to avoid collisions). This value will be used each time this peer re-publishes a content or publishes a new one.

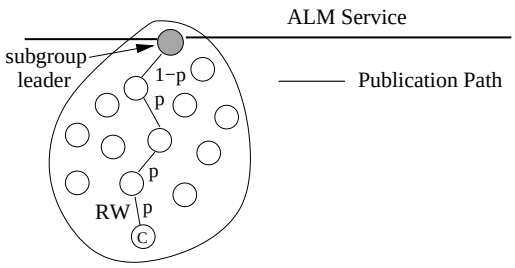

Figure 2: Publishing phase.

Figure 2 describes the random path creation proces. The content owner (*C*) randomly chooses another active peer (a peer knows all the subgroup members, which are listed in the future leader candidate list, see Section 2) and executes the RW algorithm to determine the next peer in the RW path as follows. The peer will send the *publish message* directly to the subgroup leader with probability $1-p$, or to the randomly chosen peer with probability p. This message contains the connection identifier associated to the content owner and the published content metadata. The content owner table entry associates the connection identifier and the next peer in the RW path. Each peer receiving a *publish message* follows the same procedure. It stores an entry in its table, that associates the connection identifier, the last peer in the RW path and the next node in the RW path (which will be determined by the peer using the RW algorithm as it is described before).

To prevent a message forwarding loop within the subgroup, each initial *publish message* (that is generated by the content owner) is attached with a TTL (Time To Live) value. A peer receiving a *publish message* decrements the TTL by one. Then, if the resulting TTL value is greater than 1, the peer executes the RW algorithm. Otherwise, the message is sent directly to the subgroup leader. When the *publish message* is received by the subgroup leader, it just stores the adequate table entry that associates the connection identifier, the published content metadata and the last peer in the RW path.

The probability that the RW has n hops is

$$P(n) = \begin{cases} p^{n-1}(1-p) & n < TTL, \\ p^{TTL-1} & n = TTL. \end{cases} \quad (1)$$

Therefore the mean length of a RW is

$$\overline{RW} = \frac{1-p^{TTL}}{1-p} \xrightarrow[p\to 1]{} TTL. \qquad (2)$$

Thanks to the publishing procedure, each subgroup leader knows the metadata of the contents that have been published into its subgroup. However, this publishing procedure offers sender anonymity. Any peer receiving a *publish message* knows who is the previous node. However, it does not know if this previous peer is the content owner or just a simple message forwarder.

The RW path distributely generated in the publishing phase will be used during the searching and downloading phases, as it will be described later. Therefore, it is necessary to keep updated the RW paths according to the non-voluntary peer departures. From this joint to the anonymous system, each peer maintains a *timeout timer*. When its timeout timer expires, each peer will check the RW path information stored in its table. For each entry, it will send a *ping message* to the previous and the following peer. If a peer detects a connection failure, it generates a RW failure notification that is issued peer by peer to the subgroup leader (or to the content owner) using the RW path. Each peer receiving a RW failure notification deletes the associate table entries (only one entry in intermediate peers and one entry for each published content in the subgroup leader peer). In addition, the content owner re-publishes all its contents. On the other hand, if an intermmediate peer wants to leave the system in a voluntary way, it will get in touch with the previous and the following peer associated to each table entry in order to updated their entries.

An additional verification is carried out each time a new content is going to be published. Before sending the *publish messages*, the content owner peer makes a *totalPing* (a ping sent to the subgroup leader through the RW path). If the *totalPing* fails (any intermediate peer is disconnected), the content owner peer re-publishes all its contents initiating the creation of a new distributed RW path. The intermediate peers belonging to the failed RW path will update their table entries when their timeout expires.

3.2 Searching the Contents

The searching procedure to obtain requester anonymity is described in this section. Figure 3 describes this procedure. When a requester (*R*) wants to make a complex search, first of all it must select its connection identifier. If the peer already chose a connection identifier during a publish phase, this value is used. Otherwise, a connection identifier is chosen[1].

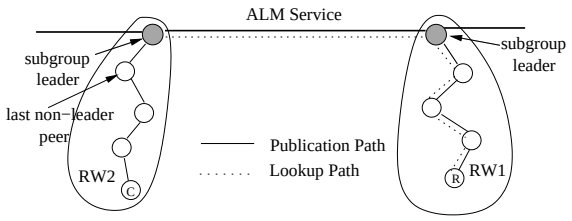

Figure 3: Searching phase.

Then, the peer generates a new type of message called *lookup message*. A *lookup message*, that encapsulates the metadata of the searched content and the connection identifier associated to the requester peer (*R*), is forwarded towards the local subgroup leader using the RW path associated to this connection identifier (RW1). If the connection identifier is new, the RW1 path from the requester peer towards the subgroup leader is generated, distributely stored, and maintained as it was explained in the section before.

Once the *lookup message* has been received, the local subgroup leader is able to locate the matching results within its subgroup. On the other hand, our system uses an ALM technique to distribute the lookup messages to all the subgroup leader nodes.

Each subgroup leader with a positive matching result must answer the requester's subgroup leader with a *lookup-response message*. This message contains the connection identifier established by the requester peer, the matching content metadata, the connection identifier established by the content owner peer and the identity of the last non-leader peer in the RW path (RW2) towards the content owner (*C*).

In fact, before sending the *lookup-response message*, the subgroup leader will check the complete RW2 path towards the content owner by means of a *totalPing*. If the RW2 path fails, the subgroup leader will send to all the peers in the subgroup a broadcast message containing the associate connection identifier. The peer that established this connection will republish all its contents (that is, it initiates the creation of a new RW for the connection identifier). Peers that maintain a table entry associated to this connection identifier will delete it. The rest of peers will take no notice of the message. Consequently, the subgroup leader only sends a *lookup-response message* to the requester subgroup leader after checking that the associate RW2 path is active.

Finally, the *lookup-response message* will be forwarded through the requester subgroup towards the requester peer using the adequate and distributed

[1]Each peer is associated to only a connection identifier.

RW1 path stored in the tables.

3.3 Downloading the Contents

Once the requester peer receives the *lookup-response* messages, it is able to select which contents it wants to download. The anonymity is also important during this third phase: both the requester peer and the content owner peer should keep anonymous to the rest of peers in the system.

In the download phase (see Figure 4), the requester peer (*R*) generates a *download-request message* which initially encapsulates the required content metadata, the connection identifier established by itself, the connection identifier established by the content owner (*C*) and the identity of the last non-leader peer in the RW2 (in the content subgroup).

This message will be forwarded towards the subgroup leader using the distributely stored RW1 path. However, this message will not reach the subgroup leader. The last non-leader peer in the RW1 path (the previous to the leader) will be the responsible for sending the *download-request message* to the last non-leader peer in the RW2 (in the content subgroup). Before sending the message to the content owner subgroup, this peer (the last non-leader peer in RW1) must encapsulate an additional value: its own identity. The message will be forwarded through the content owner subgroup using the distributely stored RW2 path until it reaches the content owner.

Finally, the content owner peer will start the content delivery. The *download messages* encapsulate the requited content, the connection identifier established by the requester peer, the connection identifier established by itself and the identity of the last non-leader peer in the RW1 (in the requester subgroup). These messages will be forwarded towards the subgroup leader using the adequate RW2 path. However, they will not reach the subgroup leader. The last non-leader peer in the RW2 path will be the responsible for sending them to the node indicated in the message. Once in the requester subgroup, the messages will be forwarded using the RW1 path until they reach the requester peer. Therefore, the mean length of a download path (DP) is

$$\overline{DP} = 2(\overline{RW} - 1) + 1 = 2\frac{1 - p^{TTL}}{1 - p} - 1 \qquad (3)$$

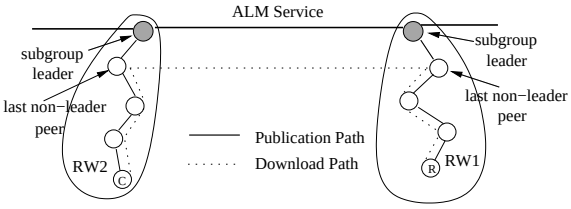

Figure 4: Download phase.

4 DYNAMIC ASPECTS OF THE SYSTEM

The peer dynamism or *churn* is an inherent feature of the P2P networks. Our solution exploits this property to improve the anonymity behavior of the system. As it has been described in the previous section, each peer establishes a RW path towards its subgroup leader during the publishing or the searching phases. If there was no churn, these RW paths (which are randomly established) would not change as the time passes. However, due to the peer departures and failures, the RW paths must be continously updated. Consequently, the RW path information that could be obtained by attackers will be obsolete as the time passes.

The random election of each RW path is another interesting feature of our proposal. The main objective of this procedure is to reach the mutual anonymity. However, there is another important consequence. The first non-leader peer (closest to the leader) in each RW path changes from a RW to another. Consequently, the remote peers involved in the download process change depending on the content owner peer and the requester peer. As a consequence, the downloading load is distributed among all the peers. And something more important, the subgroup leaders don't participate during the download process making lighter their processing load.

As it has been described in the previous sections, the subgroup leaders have a key role during the publishing and the searching phases. Each subgroup leader maintains a list of the published contents in its subgroup that is used to solve the complex lookups. The subgroup leader management is defined by the network architecture. However, there is no doubt that the subgroup leader changes also affect the anonymous system. Each time a subgroup leader changes, all the RW paths would be obtained again. To optimize the system behaviour when a subgroup leader changes we propose some actions. If a subgroup leader changes voluntarily (another peer in the subgroup has become a most suitable leader or the leader decides to leave the system), it will send to the new

leader a copy of its table. On the other hand, to solve the non-voluntary leader leaving, we propose to replicate the table of the subgroup leader to the next two peers in the future leader candidate list. Consequently, when the subgroup leader fails, the new subgroup leader will be able to update its table taking into account the published contents.

5 SIMULATIONS

We have developed a discrete event simulator in C language to evaluate our system. At the beginning, the available contents are distributed in a random way among all the nodes of the network. As it has been mentioned before, the subgroup leaders share information using an ALM procedure. The node life and death times follow a Pareto distribution using α = 0.83 and β = 1560 sec. These paremeters have been extracted from (Li et al., 2005). The time between queries follows an exponential distribution with mean 3600 sec. (1 hour), and if a query fails the user makes the same request for three times, with an empty interval of 300 sec. among them. The *timeout timer* has a value of 300 sec. Finally, the simulation results corresponds to a network with 12,800 different contents and 6,400 nodes, with 128 subgroups.

We have proposed a mechanism to provide anonymity to a hybrid P2P architecture. However, this service has a cost in terms of bandwith consumption and nodes overload. On the other hand, our anonymity mechanism also works fine under a join/leave or failure scenario. This service is required to maintain available paths along the time, but it involves an extra control packet interchange. It is necessary to quantify both costs.

Figure 5 represents the average number of hops in a complete download path, from the content owner to the requester node (download path length). This measurement helps us in order to estimate the bandwith consumption, since every hop in a path implies an extra bandwith consumption to carry out the download. This figure represents the results in function of p and TTL. We represent the results for 3 different TTL values (10, 15 and 20). When p is less than 0.8 the mean path length never exceeds 7, so the extra bandwith consumption is very limited. In addition, in this case the TTL guarantees that in any case the number of hops will be limited. If p is greater than 0.8 the extra bandwith consumption tends to infinity, but the TTL limits it to a reasonable value.

This simulation result corresponds with the analytical expresssion presented in Equation 3. Additionnaly, we represent the number of hops in a compelte download path without using a TTL limit. We observe than if p is greater than 0.9 the extra bandwith goes up quickly.

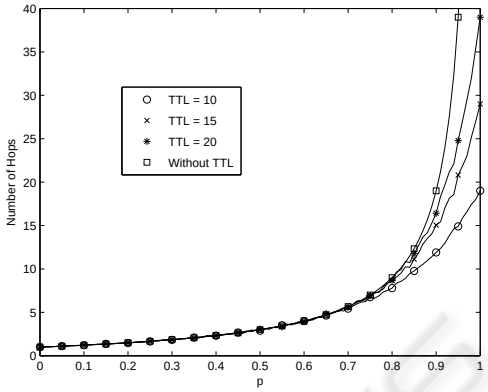

Figure 5: Average number of hops in a complete download path.

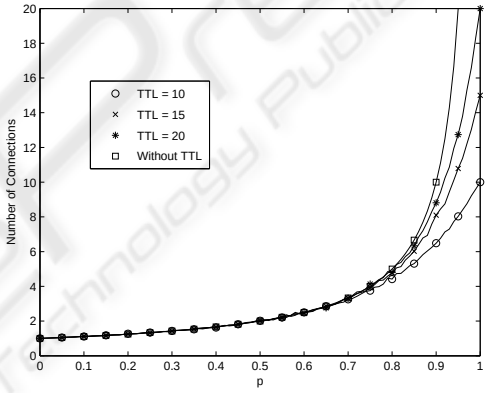

Figure 6: Average numer of connections that a node routes.

Figure 6 represents the average number of connections that a node routes in function of p and TTL. With p = 0 the number of paths is 0, because the connection is established directly between the owner and the requester node. If the value of p increases, the number of connections also increases. The influence of TTL doesn't appear until p = 0.8, as we observed in Figure 5. When $p > 0.8$ the number of connections a node routes does not tend to infinity thanks to the TTL.

This simulation result corresponds with the analytical expresssion presented in (Sui et al., 2003). Additionnaly, we represent the the average number of connections that a node routes without using a TTL limit. We observe than if p is greater than 0.9 the number of connections goes up quickly. As a conclusion, a value of p = 0.8 is a trade off solution

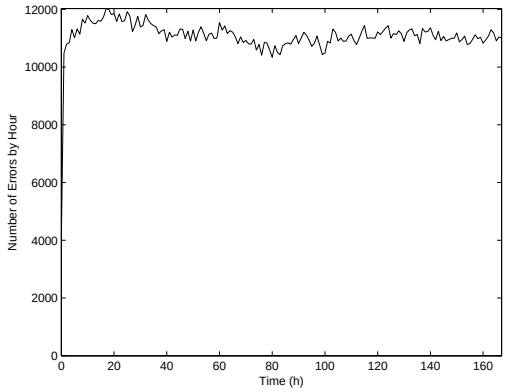

Figure 7: Number of request that cannot be carry out in an hour because at least one node in the download path is out.

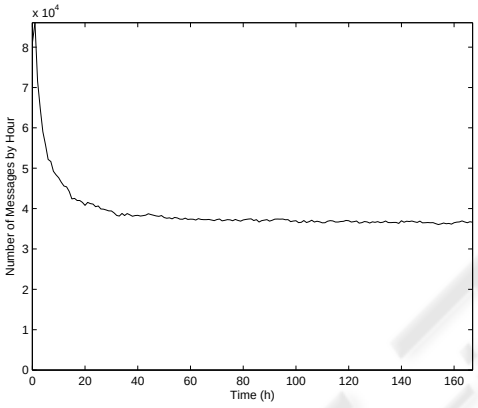

Figure 8: Average number of control messages in function of time.

between anonymity efficience, extra bandwith consumption and nodes overload.

In the results represented in Figure 7 our system doesn't implement the reliability mechanism. The parameters used in the simulatons corresponds to a typical scenario: $TTL = 10$ and $p = 0.7$. It shows the number of requests that cannot be carry out in an hour because, at least one node in the download path is down when the download is performed. This result is represented in funcion of time (in hours). This number fluctuates around 11,000.

Therefore, it is clear than in a real anonymous P2P network (with unpredictable node failures) it is necessary to suply a mechanism to reconstruct efficiently the paths, although it entails an extra network overhead in terms of control packets.

In our network, a download process never fails since it is a reliable anonymous P2P network, but this feature involves an extra control traffic. Figure 8 represents the average number of control messages in function of time, in the previous scenario. Initially, due to simulation constraints, the *timeout timer* expires simultaneously in a lot of nodes, but in steady state the average number of control messages is under 40,000.

Usually, control messages have a small payload (about 10 bytes) and a IP/TCP header (40 bytes). If we suppose each control message has a length of 50 bytes, control traffic supposes only a 4.4 kbps traffic rate.

6 CONCLUSION

In this paper, we have proposed anonymity mechanisms for a hybrid P2P network presented in a previous work. Unfortunately, the mechanisms to provide anonymity entails extra bandwidth conssumption, extra nodes overload and extra network overhead in terms of control traffic. The simulations tries to evaluate the costs to provide anonymity in a real P2P scenario (with unpredictable node failures). As a conclusion, our proposal achieves mutual anonymity only with an extra bandwith consumption corresponding to 5 hops, 3 connections by node and a 4.4 kbps control traffic rate (in a typical scenario $TTL = 10$ and $p = 0.7$).

ACKNOWLEDGEMENTS

This work has been supported by the Spanish Researh Council under project ARPaq (TEC2004-05622-C04-02/TCM). Juan Pedro Muñoz-Gea thanks the Spanish MEC for a FPU (AP2006-01567) pre-doctoral fellowship.

REFERENCES

Castro, M., Druschel, P., Kermarrec, A., and Rowstron, A. (2002). Scribe: A large-scale and decentralized application-level multicast infrastructure. *IEEE Journal On Selected Areas in Communications*, 20(8):100–110.

Dingledine, R., Mathewson, N., and Syverson, P. (2004). Tor: The second-generation onion router. In *Proceedings of the 13th USENIX Security Symposium*, San Diego, CA, USA.

Freedman, M. and Morris, R. (2002). Tarzan: A peer-to-peer anonymizing network layer. In *Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS'02)*, Washington, DC, USA.

Ghodsi, A., Alima, L., El-Ansary, S., Brand, P., and Haridi, S. (2003). Dks(n,k,f): A family of low communication, scalable and fault-tolerant infrastructures for p2p applications. In *Proceedings of the 3rd. International Workshop on Global and P2P Computing on Large Scale Distributed Systems (CCGRID 2003)*, Tokyo, Japan.

Han, J., Liu, Y., Xiao, L., Xiao, R., and Ni, L. M. (2005). A mutual anonymous peer-to-peer protocol design. In *Proceedings of the 19th International Parallel and Distributed Processing Symposium (IPDPS'05)*, Denver, CO, USA.

Levine, B. N. and Shields, C. (2002). Hordes: A multicast-based protocol for anonymity. *Journal of Computer Security*, 10(3):213–240.

Li, J., Stribling, J., Morris, R., and Kaashoek, M. F. (2005). Bandwidth-efficient management of dht routing tables. In *Proceedings of the 2nd USENIX Symposium on Networked Systems Design and Implementation (NSDI'05)*, Boston, MA, USA.

Lu, T., Fang, B., Sun, Y., and Cheng, X. (2004). Wongoo: A peer-to-peer protocol for anonymous communication. In *Proceedings of the 2004 International Conference on Parallel and Distributed Processing Techniques and Appliations (PDPTA 2004)*, Las Vegas, NE, USA.

Mislove, A., Oberoi, G., A. Post, C. R., and Druschel, P. (2004). Ap3: Cooperative, decentralized anonymous communication. In *Proceedings of the 11th workshop on ACM SIGOPS European workshop: beyond the PC*, New York, NY, USA.

Muñoz-Gea, J. P., Malgosa-Sanahuja, J., Manzanares-Lopez, P., Sanchez-Aarnoutse, J. C., and Guirado-Puerta, A. M. (2006). A hybrid topology architecture for p2p file sharing systems. In *Proceedings of the First International Conference on Software and Data Technologies (ICSOFT 2006)*, Setúbal, Portugal.

Pfitzmann, A. and Hansen, M. (2001). Anonymity, unobservability and pseudomyity: a proposal for terminology. In *Proceedings of the Fourth International Information Hiding Workshop*, Pittsburgh, PE, USA.

Ratnasamy, S., Francis, P., Handley, M., Karp, R., and Shenker, S. (2001a). A scalable content-addressable network. In *Proceedings of ACM SIGCOMM*, San Diego, CA, USA.

Ratnasamy, S., Handley, M., Karp, R., and Shenker, S. (2001b). Application-level multicast using content-addressable networks. In *Proceedings of the 3rd. International Workshop of Networked Group Communication (NGC)*, London, UK.

Reed, M. G., Syverson, P. F., and Goldshlag, D. M. (1998). Anonymous connections and onion routing. *IEEE Journal on Selected Areas in Communications*, 16(4):482–494.

Reiter, M. K. and Rubin, A. D. (1999). Crowds: Anonymity for web transactions. *Communications of the ACM*, 42(2):32–48.

Rowstron, A. and Druschel, P. (2001). Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems. In *Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, Heidelberg, Germany.

Stoica, I., Morris, R., Liben-Nowell, D., and Karger, D. (2003). Chord: A scalable peer-to-peer lookup protocol for internet applications. *IEEE/ACM Transactions on Networking*, 11(1):17–32.

Sui, H., Chen, J., Che, S., and Wang, J. (2003). Payload analysis of anonymous communication system with host-based rerouting mechanism. In *Proceedings of the Eighth IEEE International Symposium on Computers and Communications (ISCC'03)*, Kemer-Antalya, Turkey.

Wright, M., Adler, M., Levine, B. N., and Shields, C. (2002). An analysis of the degradation of anonymous protocols. In *Proceedings of the Network and Distributed Security Symposium (NDSS'02)*, San Diego, CA, USA.

Xiao, L., Xu, Z., and Shang, X. (2003). Low-cost and reliable mutual anonymity protocols in peer-to-peer networks. *IEEE Transactions on Parallel and Distributed Systems*, 14(9):829–840.