Projecte de recerca bàsica o aplicada
PAC3 — Tercera Prova d'avaluació continuada

Cognoms: *Sanchez Cañadas*
Nom: *Oscar*

- Per a dubtes i aclariments sobre l'enunciat, adreceu-vos al consultor responsable de la vostra aula.
- Cal lliurar la solució **en un fitxer OpenDocument o PDF fent servir una de les plantilles lliurades conjuntament amb aquest enunciat. Adjunteu el fitxer a un missatge adreçat a l'espai d'avaluació de l'aula virtual**.
- El fitxer ha de tenir l'extensió *.odt* (OpenDocument) o *.pdf* (PDF) segons el format en què feu el lliurament.
- La data límit de lliurament és el **19 de gener** (a les 24 hores).

Respostes

Anonymous Routing Protocol in decentralized networks

Universitat Oberta de Catalunya (UOC)
Author: Sanchez Cañadas, Oscar. Student at the UOC and Telecommunications Engineer
Tutors: **Rifa, Helena & Marquès, Joan Manuel.** Faculty of computer science and multimedia
January 2012, Barcelona, Catalonia, Spain

ABSTRACT

We propose an algorithm for performing routing anonymous within the networks peer-to-peer, decentralized networks.

We have created a small application for sending a file from a user who connects to the application to a server that is assigned to it, which is given to a Directory, which contains all the addresses of these Entities.

We have also done a few tests to check the behavior of the algorithm within an application, such as the degree of anonymity provides, or latency of this depending on the size of the file or the performance of each Entity by package.

And finally we extract conclusions and propose improvements to the algorithm for future works.

1 INTRODUCTION

In the last decades the widespread use of the Internet has increased the interest on methods for the protection of privacy and communications, both from the academic community and the general public.

Several system designs have been proposed in the academic field, some of which have been implemented and are used by various groups in order to protect your identity on the Internet. Although encryption can hide the contents of the information in the network, it is not enough to hide the IP address from other users.

The architecture of the decentralized systems do not require a centralized server. This design can prevent many attacks that are made to a centralized server, as no attack can break all nodes. Besides this type of architecture is able to make the balance of the load by itself.

Our goal is to design and implement a robust system, high-availability with free storage content, and Furthermore, it must guarantee the reliability, privacy and anonymity.

2 JUSTIFICATION

Currently, a large number of P2P platforms, distributed systems and shared storage mechanisms have adopted research-based routing and distributed hash tables (DHT). DHT provides a system of self-organization with high routing performance, searching precision, high scalability and automatic load balance.

DHT can be built on a network of volunteer teams comprised of end-users to supply their own. The simplicity and flexibility to become a member of volunteer network in these networks gives a big potential computational and storage. However, this kind of network also face a big quantity of security problems. All peers of these networks can transmit, receive and forward data, so we become very vulnerable, because any peer can receive the information from other peers and may have malicious peers, infected content, spy communications and many problems related with security. Currently there are many problems in such networks, which could be mitigated with a protocol to provide anonymity.

Therefore, in this project we focus on reliability and privacy issues related to storage of data. The research support is based on the design and the creation of a protocol that allows us to guarantee the anonymity of volunteers within a distribution network based on DHT.

3 OBJECTIVES OF RESEARCH

The aim of this project is the design and implementation of a robust system, highly scalable and content-free to resist the attempts of powerful adversaries lock. All content can be stored and searched publicly and is available in the most convenient possible way.

Anyone can publish any content stored on the network, while avoiding the removal of the content or the determination of the true identity of the people involved by internal or external adversaries. In addition, the peers members that contribute to the storage ought not be an easy way to determine what they are storing, ensuring a similar denial.

In summary, the primary concern of this design is the creation of a proposed protocol for anonymity to provide privacy to the content stored by volunteers, ensuring their anonymity and confidentiality, avoiding any impersonation.

4 CURRENT OVERVIEW OF THE ANONYMITY SYSTEMS

We analyze the algorithms based on low latency because our research is based on these types of systems.

4.1 Onion Routing

Onion routing is design with a higher prevalence for anonymous communications. Each onion router maintains a private key and public. The public key has to inform and to know the customers. To set the anonymous connections initiator builds a multiple encryption tunnel, or circuit, through the network. All processed messages must be encrypted asymmetrically.

Onion routing senders on a network using only the public key to establish a circuit of encryption, and then use the cryptography through symmetric key to transfer faster the real data.

The initiator generates two symmetrical secret keys, to re-send and another to answer. The initiator sends the first onion router in the path, which eliminates the outermost layer of the encryption with its private key. Each server along the route will repeat the same operation until the onion reaches the end of its route. Once built the circuit, the initiator can retransmit traffic using the symmetric key generated on each hop. [1]

4.2 PipeNet

The algorithm selects a random sequence of PipeNet servers network, similar to the route selection of onion routing. Customers then create multiple encrypted tunnel via the establishment of a symmetric key with the first hop, a tunnel through the encrypted connection and the establishment of a leap second, and so on. [1]

Although the design of PipeNet is impractical for real networks like the Internet.

4.3 Tarzan

Tarzan, AP3 and Crowd are very similar systems of anonymity.[3.5]

The initiators in Tarzan build a network of routes through this by generating symmetric keys for each hop and encrypting them with public keys of servers in the circuit.

As the Crowd system, all participants in the network broadcast by other users.

Tarzan is a flexible, transparent, decentralized and highly scalable system. Use tunnels between the nodes and is transparent to both the client and to the server. This system is able to provide anonymity to existing applications without making any modifications. [6]

Tarzan uses a peer-to-peer protocol gossip in order to share information on all servers. Discover the servers by selecting a random node in the residents who already know , like AP3 and Crowd.[5]

The initiator of an anonymous tunnel creates a circuit through the network Tarzan is grabbing randomly this first. The second step is selected via the set chosen by the first hop, and so on. Application traffic is sent encrypted through the circuit with a layered, very similar to what we have already discussed with the onion routing. [1]

4.4 MorphMix

MorphMix is also similar to Tarzan, is a system of low-latency anonymous peer-to-peer where all network nodes are of re-transmitter. This also uses multiple layers in order to make the symmetric encryption before transferring data along the circuit.

A node establishes a symmetric key by sending ab half of the algorithm encryption with Diffie-Hellman public key witness w. The node b transmits the encryption to aw that removes the layer of encryption and sends the result to a. This generates half of the Diffie-Hellman algorithm and sends it through b. Both parties are able to generate their shared symmetric key. [1]

4.5 Salsa

Salsa is a system based on anonymous DHT layer. Each node has a specific ID to perform the cryptographic hash function to obtain the IP address. [3]

The recursive searches to receive a public key and the keys of each re-transmitter are performed by initiator’s request of r nodes including it himself. The

circuit will spread randomly.

If a result of research is greater than a threshold distance to the destination ID, it will be discarded.
The initiator uses a cryptographic hash function to calculate the public ID of each IP address back. The ID of the closest will be the destination key selected, and the other results are discarded. [3]

The test limits will not be effective in case the attacker can make the control key of the closest node. [3]

4.6 Cashmere

This system uses virtual re-transmitter made up by a set of nodes for the resistance. [3]

Each group of re-transmitter must have a public/private key pair, and each Member of the group that has had a public/private group.

The group ID will be used as the key for sending messages, using a prefix. The root of the group is responsible for processing messages in number of re-transmitter. To detect faults and malicious nodes using end-to-end, Cashmere ACKs. [3]

4.7 Tor

Tor is the second generation Onion Routing System created by the limitations of the original design, adding direct perfect confidentiality, congestion control, directory servers, integrity check, Out-effective set of policies, and a practical design for localization services through meeting points. [4]

Tor provides confidentiality, as it now uses an incremental design for the construction of the path, where the initiator negotiates session keys in each hop on the circuit.

4.7.1 Improvements with respect to Onion Routing

Is more reliable, since the initiator knows when a hop fails and you can try to extend a new node.

Improves efficiency and anonymity in the multiplexing TCP flows in each circuit.

The initiator can send traffic to the nodes up to half of the circuit, avoiding thus the attacks based on the observation of the end of the circuit.

It incorporates a congestion control. End-to-end by ACK can maintain anonymity while these nodes allow to detect congestion and to control it.

Tor provides Directory servers signed to discover the known nodes and their current States.

Incorporates an end-to-end integrity check, so that any node can change the information content of the cells.

Provides a point of rendezvous and hidden services. Customers negotiate the rendezvous point to connect to hidden services. [4]

4.7.2 Directory Servers

Directory servers are responsible for the aggregation and distribution of the information registered in the network of known routers. Information of signed directory may also be reflected by the other routers in the network in order to reduce the load on the servers Directory. [1]

The use of these servers is simpler and more flexible than using flooding. [4]

4.7.3 Establishment of the circuit

The circuit initiator negotiates session keys with each router in the path of the circuit through the negotiation of DH and RSA authentication. [1]

Face onion router keeps a key identity that is used to sign certificates, TLS. The onion key is used to decrypt requests from users for the performance of a circuit and the negotiation of keys. [4]

4.7.4 Location hidden services

The location of hidden services allow offer to TCP services, as a Web server, without the need to reveal the IP address. This type of anonymity protects against DoS attacks, since the IP is not known.

4.7.5 Solutions to the problems of current Tor

To solve the problems of scalability in the Tor network, we propose two systems, NISAN and Torsk. The designs including mechanisms in order to mitigate the attacks leak of information. Nisan proposes to incorporate anonymity in their own research and instead of Torsk uses nodes buddy. [5]

4.8 NISSAN

Nisan proposes a construction with 3 re-transmitter random from a list that provides the router to build the tunnel. This configuration could allow an attacker to link the output of the last node to the initiator, as the last node is in direct contact with the initiator, so if this node is compromised, the attacker can break the tunnel. [5]

To avoid this we can establish a partial circuit of the

first node and after with a next search the second node after the third, and so on. Unfortunately this construction by itself is vulnerable to attacks public-key modification and route capture attack. [5]

One way to mitigate the attacks of the first construction is to use a longer route. Although this construction could increase the latency of booting system and makes the system more vulnerable to DoS attacks. [5]

4.9 Torks

Torsk requires the selection of a random number of nodes across the all network to select privacy. An attacker cannot associate the objective in order to research the query. Therefore, the following hop-by-hop may not be applied and the buddies are kept secret. [5]

Torsk proposes to use a random path for the selection of buddies. After the process buddy is performed off-line. Then Torsk request which buddies will be used one time.

Finally Torsk Finally, the veracity of the certificate applies Torsk each hop of random path, preventing further attacks that has random paths to increase the chance of malicious nodes that are selected as buddies.

5 ANALYSIS of ANONYMOUS P2P SYSTEMS

5.1 System Structure

The decentralized architecture of P2P systems can keep more peers of users and services than can maintain a centralized architecture. Therefore, the decentralized architecture is much more scalable and the scalability of the network is a very important element for any system.

If the radius of malicious peers is unchanged, the degree of anonymity of the system increases with increasing the scale of the network, because if the number of malicious peers is unchanged, and the network grows with honest peers, the proportion of malicious peers will become increasingly small and therefore the degree of anonymity will be greater. Thus, if malicious peers increases, the degree of anonymity will suffer a decrease. The P2P network is ideal for building a large network.

The degree of anonymity seems relatively high even if the network environment is very negative. This means that if all the malicious peers can see only the communication, but can not be part of it and all the communication features of honest peers would look like equal, the system can also maintain a high degree of anonymity.

When a peer needs a service anonymous, the first thing to do is looking for peers through DHT. This may be a victim of the leakage information in anonymous communications. If a user gets to interfere in this process, user will be able to find the transmitter, but can not identify the tunnel that transmitter created. Also if the last peer are malicious, the real target may be identified, as the attacker can trace the real sender of this tunnel using the timing attack or bridge attack.

The difference with the structured P2P systems is that they use the gossip protocol to discover active peers and select the re-transmitter. These systems are less affected by malicious peers than unstructured systems.

5.2 Size of the tunnel

In the process of routing, each peer only knows the re-transmit its predecessor and its successor, so the attacker needs at least control more than half of the peers in the tunnel for its reconstruction.

At first, the degree of anonymity increases rapidly with increasing size of the tunnel, but when the size exceeds more than 10 hops, the degree of anonymity begins to decrease slowly. This is why, because a tunnel is very wide and it is very difficult to control the attacker, the more we add hops, give the attacker more chances to join our tunnel and find the sender. [2]

The variable size of the tunnel can provide a higher degree of anonymity that a fixed size.

5.3 Selection of peers re-transmitter by merging of cluster

With the Merger of cluster method, the degree of anonymity increases when a cluster grows with the scale of the network, but is still relatively small. The main reason for this is because the cluster is much smaller than the total number of network peers and clusters do not cooperate with each other, so anonymity is smaller. If we combine the clusters, and if the opponent wants to break the tunnel, this will have to join them. So the strategy is that only the sender selects the next peer to be preferable in design system.

5.4 Relay Strategy

The relay strategy which is used the most in systems is onion routing. When the message is relayed from one peer to the next peer, that message is encrypted with the symmetric key of the next peer, and the next

peer decrypts the message and encrypt it again with the key of the next peer and it is sending it to the next peer, and is continuing the same process until reaches the receiver peer. Therefore, if it is a malicious peer, it must control the first and the last peer if you want to analyze the messages inside the tunnel.

If the first peer of the tunnel is malicious, it can identify the relationship between you and the real server. If the first honest peer, another peer in the tunnel can not identify the sender directly and there is only one real chance 1 / N (N total number of peers) to determine the correct identity of the sender. [2]

6 ANONYMOUS ROUTING PROTOCOL

We built a decentralized application to communicate us with a server, so that this communication is anonymous and there is no central server to manage information.

6.1 Application to decentralized anonymous communications

In this application we created an interface to the user (User Application) so that this can send documents to the server. This interface makes the transmission through the anonymous routing protocol that we created.

6.2 Services Entity

We have created Entities, which are responsible to retransmit packets and manage the documents that users send. Each Entity is replicated in several instances to ensure availability of information in case of failures.

6.3 Directory Servers

We have created a Server Directory, which is responsible of launching the Entities and to assign it to users.

6.4 Certificate authority

Creating a certificate authority (CA) was required to certify the identity of users and services involved in the publication of the document.

It creates a self-signed certificate for both CA and for the Entities and get within a PKCS#12 key repository, where it save it also certificates of the users and the Entities.

6.5 Functionality

The user enters their work to the User Application. It contacts with the Directory through a SSL connection with mutual authentication to obtain the ID of the Entity where they will work as shown in Figure 1.

The Directory assign a free Entity to the user, and marked this Entity as "in progress", and will return a list of Entities and their destiny Entity.

Then User Application execute the Anonymous Routing Protocol by setting as destination the Entity ID of the origin Entity that is taken at random from the entire list of Entities.

The destiny Entity takes document and stored, instead, although this does not send any confirmation to the user, will be the origin Entity, after the destiny Entity appears, the responsible of communicate to user the result of that operation. Therefore, the User Applications will display the result to the user.

Finally, after do the operation and the User Application receives the confirmation, the Directory will mark the Entity as "free".

6.6 Algorithm of Anonymous Routing Protocol

The Anonymous Routing Protocol is based on Onion routing where a message is encrypted in multiple layers. Each layer contains the identity and a packet encrypted for the next router on the network.

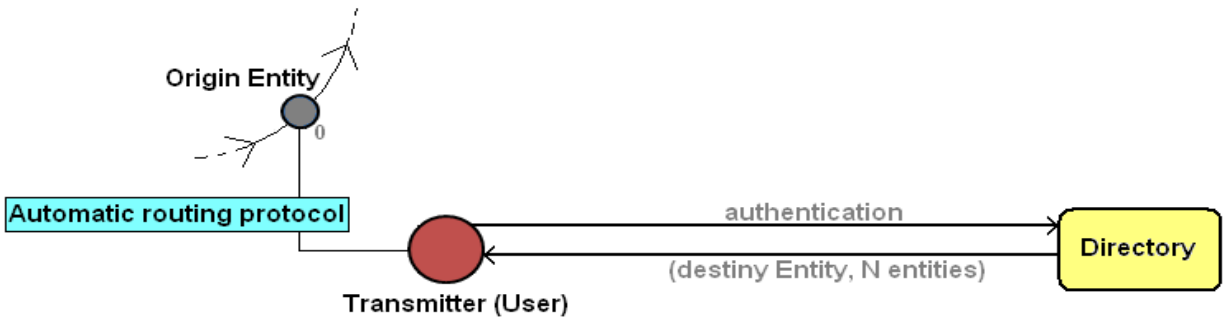

Figure 1. Entity Authentication and assignment destination

6.6.1 Tunnel Size

At start up the application the application in order to upload a document to the server, the Directory creates *N* random identifiers, with which the User Application creates a chain of M Entities, which is the size of the tunnel used in the protocol.

As the authors, Jia Zhang, Jianping Wu, Duan, Liu Wu Haixin and indicate to their article [2] *Anonymity analysis of P2P anonymous communication system*:

> "At the beginning, the anonymity degree rises rapidly with the tunnel length increasing, but when the length exceeds 10, the anonymity degree will not rise, but instead declines slowly. This is because although a long tunnel is hard for adversary to control, it also gives adversary more chances to join in and guess the sender. This phenomenon means frequent weak attacks can also make the system fragile. So, how to make tradeoffs to avoid frequent weak at- tacks should be considered by developers."

They analyzed different sizes and they saw that with a small tunnel size, the system was giving easily to the attacker to know who was the transmitter, and although the tunnel was too large, the degree of anonymity came to a point where I could not grow further because when the tunnel increase, it increase the chance that an attacker form part of the tunnel.

Therefore, after analyzing the situation, the author proposes that the ideal size of the tunnel is 10, although they also say that a variable size tunnel can provide greater degree anonymity to the system than a fixed size.

Therefore, in our protocol, we have created a variable tunnel size, where it is determined by the number of users.

$$N = (A^{1} * 2) + 10$$

Thus we improve the degree of anonymity by ensuring that there are a minimum number of 10 Entities deployed, and that the ID's will not be finish due to failures during the process of the publication of documents, since there are at least 2 more Entities for each user.

6.6.2 Selecting the nodes of the tunnel

The Anonymous Routing Protocol creates a chain of M Entities with the N Entities that created the Directory.

$$M \geq (N + 2)$$

The protocol builds the chain of M Entities sorted by random such that the destiny Entity is in the midst of this chain, and ensuring that the origin Entity, the Entity located in the first place, it becomes a repeat at least once more.

The destiny Entity precedes the second appearance of the origin Entity, in order to be the origin Entity who responds to the user about the resolution of the transmission. The position of the second origin Entity is random, it only has to appear between destiny and last Entity.

In order to prevent the origin Entity is the last Entity, we create an instance of an Entity when the origin Entity is in the last position.

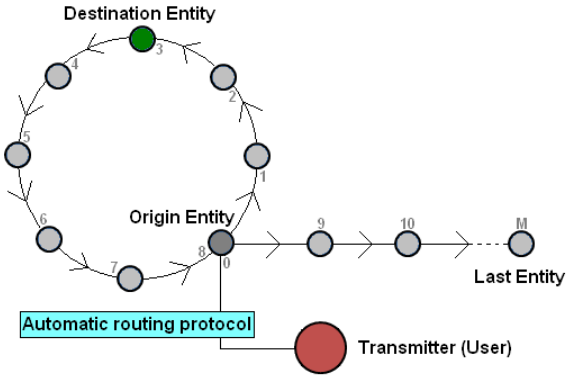

Figure 2. Example of a chain

As we see in Figure 2, the origin Entity is at position 0 the first time, and the second time is at position 8, which sends the result of the transmission to the user.

6.6.3 Creating packet to send

Once the user already has authorization by the Directory, this builds the package to send the data to the destiny Entity.

1. User Application generates M symmetrical keys associated with each of the Entities in the chain (Figure 2).

2. User Application builds the package to send the message that will be sent beginning with last Entity in the chain::

$$Packet_M = Id\_Entity_M,\ E_{pbk\_M}(K_M, 0),\ padding$$

As you can see, the first package is composed by the last Entity of chain ID (Id_Entity_M), and the symmetric key (K_M) generated by the user, and encrypted with the public key of the Entity, and finally the optional padding.

3. User Application starts to nest packages depending on the corresponding Entity:

1 A: Number of users in system

a) If next Entity is a relay Entity, the algorithm will create the following package:

$$Packet_X = Id\_Entity_X,\ E_{pbk\_X}(K_X, len_1),\ E_X(role, Id\_Entity_{X+1},\ Packet_{X+1}),\ padding$$

As we observe, the package which is created for this Entity owns the package ($Packet_{X+1}$) and the ID (Id_Entity_{X+1}) of the Entity that will go after. Also in this case we have the *role* parameter that will indicate the action of this Entity, in this case "router", since it only is a Relay.

You can also see how inside the encrypted data with public key of symmetric key is encrypted with len_1 that also serves to determine where exactly the size of encrypted data, since during the retransmission of messages, the Entities between the tunnel can add padding to packets in order to alter their appearance and increase anonymity.

b) If the next Entity is an origin Entity, algorithm will create the following package:

$$Packet_X = Id\_Entity_X,\ E_{pbk\_X}(K_X, len_1),\ E_X(role, messageId,\ Id\_Entity_{X+1},\ Packet_{X+1}),\ padding$$

This package to the origin Entity has the *role* as a "origin", parameter which indicate to the Entity that receive the message that this has been created by itself, and it has internal ID as *messageId*.

c) If the next Entity is the destiny Entity, the algorithm will create the following package:

$$Packet_z = Id\_Entity_X,\ E_{pbk\_X}(K_X, len_1),\ E_X(role, authz,\ op,\ len_2,\ \langle data\rangle,\ Id\_Entity_{X+1},\ Packet_{X+1})$$

When the package for destiny Entity is create, we put the role as a "destiny", and also brings an operation code *op* to determine the operation to do, send or receive a document. It also is composed of *data* and len_2 parameter which is the size of data. We also incorporated the term *authz*, which is the authorization code of the operation.

4. Finally, we send the following message:

$$Packet = Packet_1,\ EC,\ padding$$

Where we see that the final package incorporates the package of origin Entity, EC, that is an error code of 1 byte, and the padding

6.6.4 Relay Strategy

Once the origin Entity receives the package sent by the User Application, this resend the package to the next Entity and sets a timeout for if some error occurs.

In the case of some error appears during the retransmission and the origin Entity not received the packet to retransmit again, or the *messageId* is unknown, the origin Entity would shut down the connection with the user informing him the corresponding error code.

When a Entity receives a new packet, decrypts the encrypted text message using the private key and verify:

a) If $len_1 = 0$, is the last Entity, node of the end of transmission and should not retransmit any packet.

b) If *role* = "origin", is the origin Entity, thus it relays the package and starts the timeout. If the timeout is finish, it will send to the user a package with *EC* = 2. If however, is the second time it appears, will be sent confirmation to the user with the *EC* set for the destiny Entity.

c) If *role* = “router”, is a relay Entity and thus be retransmitted the following package.

d) If role = “destiny”, is the destiny Entity, so this checks that the user has permission to perform the operation, and if that is not authorized, the error code *EC* = 7 will be sent to the user. Then the Entity will store the content of <*data*> if $len_2 \neq 0$. If the operation went well is set EC = 1.

When User Application receives the confirmation message closes the connection and transmission and the message is displayed to the user.

7 ANALYSIS OF ANONYMOUS ROUTING PROTOCOL

To make the analysis we change the padding of the packages of the Entities. Once we have defined the padding we go variant the timeout of the connection between the origin Entity and the user.

All the tests that will be showed below were made through writing logs within the protocol, in order to test it. And also all the tests were carried out on *localhost*.

7.1 Padding Test

To make an estimate of the padding most efficient to our protocol we did several tests varying the size of the padding. Fixed a timeout on the connection between the origin Entity of the second time and the user.

To begin we posted a padding to all Entities of 5 KB size in order to see the resources that it consumes

each Entity and check the size of the packages that come from each of the Entities.

7.1.1 Fixed padding to 5KB size

We tests of padding with a document of size 1 MB and check processing time of total transfer, and the processing time for each Entity according to the variation of users, and thus the variation of the number of Entities in the tunnel.

The first test is done simulating that there is only one user on the system, therefore there is a minimum number of 12 Entities.

With this system that is using a tunnel of 12 nodes at least, an attacker that break a node, if it is not the first node, it will not be able to know who is the source of this package, and will not be able to know if the sender was the previous node, since the package is encrypted with the symmetric key that is encrypted with a RSA public key.

If the malicious node is the first, could discover the sender node although if the first node is honest, no other node of the tunnel will be able to identify the sender node directly.

The same applies to the role of the Entities, an attacker breaking a node may not know who is the source, origin, destiny or relay node.

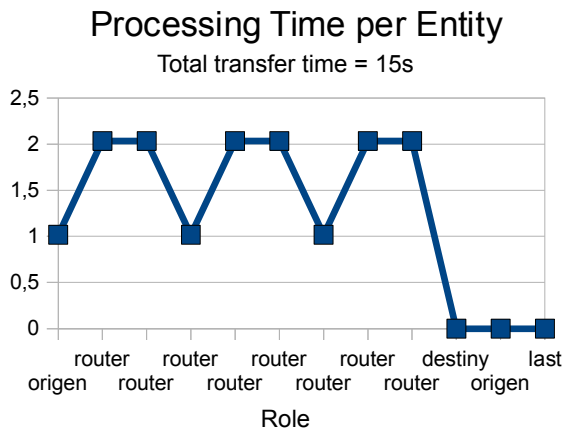

Figure 3. Processing Time per Entity

After launch, we see that 12 Entities appears in the system, and that the total transfer time of the document was 15 seconds, an acceptable time to transfer a file of 1 MB.

In Figure 4 are displayed various tests analyzing the time of transmitting 1 MB variant the number of users on the system, from 0 to 5, and therefore the number of Entities.

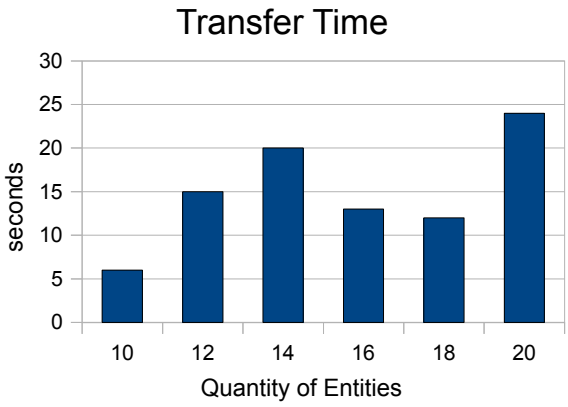

Figure 4. Transfer Time

As we can see, as more Entities in the system, more time for the transfer of the document, although it depend on the position in which appears the destiny Entity, since if it appears in the beginning, the transmission will be faster, not all Entities have to transmit the file, only those who are before of the destiny Entity. And as we observe in Figure 4, in tests where the quantity of Entities in the system were 16 and 18, the transmissions were faster, because the destiny Entity has appeared before in that two tests, and less nodes have had to carry the file.

In our test the destiny Entity appeared in the antepenultimate position, it means that the 9 previous Entities have had to transmit a packet larger than 1 MB, and thus have needed more time.

If we look at Figure 5, we see the size of the packages in each Entity.

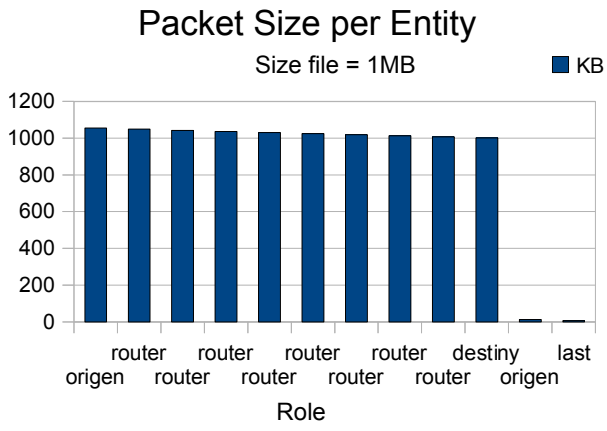

Figure 5. Packet Size per Entity

As we can see in the graph of the size of the packages that arrive in each Entity, we see a clear decrease in the size of packages after destiny Entity download the document.

Therefore, even if an attacker breaking a node might

not know who is the source of the packet, an attacker who gets control the entire tunnel can determine the trace followed by the package, since we can see that the size of packets each time it pass an Entity was decreasing, and an attacker can assume that the destiny Entity is one that takes more data of the package, observing the decrease in the size of the package once after this.

If we look at the difference with the size of the packages between Entities, as shown Figure 6, we can see clearly which Entity is getting more data.

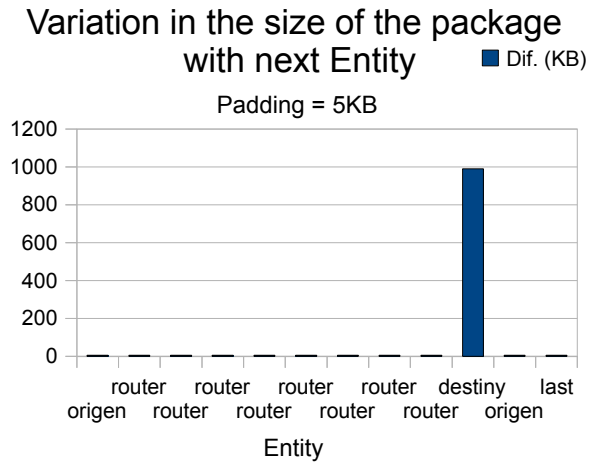

Figure 6. Variation in the size of the package with next Entity

To solve these problems, we propose to add a padding to all Entities of the same size as the file, this way an adversary can not find the destiny Entity.

7.1.2 Padding equal to file size for all Entities

In that case we repeat the test with a 1MB file and one user, that is, at least 12 Entities, but the padding that use is as big as the file size to send to the destiny.

In this way we prevent that an adversary can control the entire tunnel to find out who is the destiny Entity.

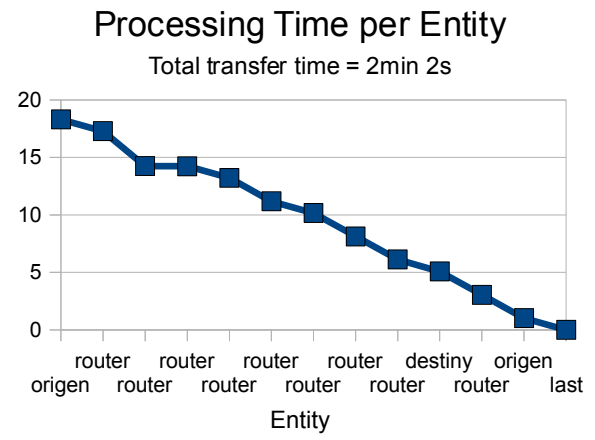

Figure 7. Processing Time per Entity

If we look at Figure 7, we see that the processing time needed for each Entity in system.

As we can see, this time have been appear 13 Entities, and we can observe the processing time of Entities decreases every time you pass one of them, since the package to process each is getting smaller.

Because of this, the time needed to transfer it by the system was 2 minutes and 2 seconds, a latency higher than in the previous case, but this system gives us more anonymity in the system.

In Figure 8, we can see the differences between the packages arrived at the Entities in this test, with the padding equal to the size of the file.

This time an adversary can not guess who is the destiny Entity, because all Entities are taking a nearly identical package.

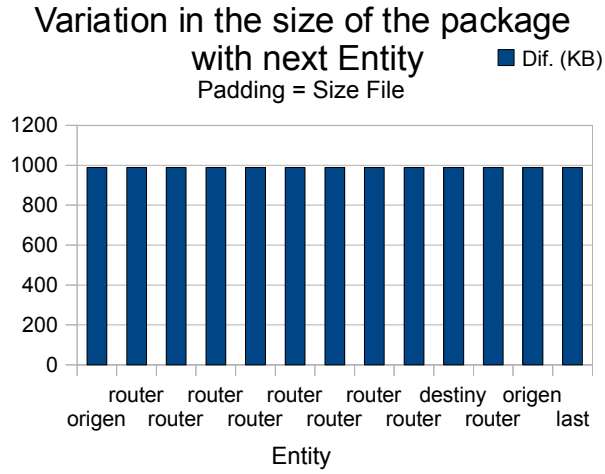

Figure 8. Variation in the size of the package with next Entity

But if we look at Figure 9, which shows the size of the package that arrives at each Entity, we can see how the packet size decreases.

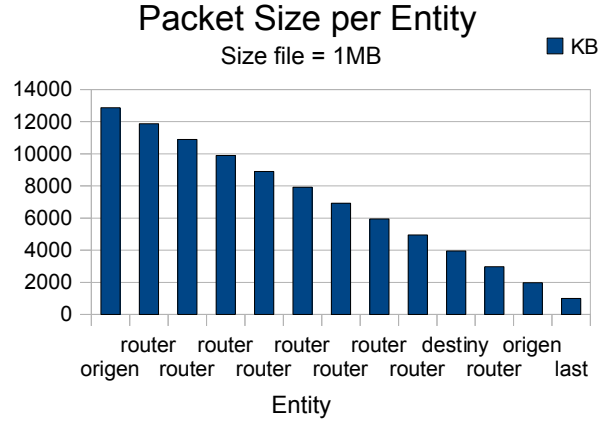

Figure 9. Packet Size per Entity

In this note, an attacker could guess the package trace and more could know who is the origin Entity, if it controls all the nodes of the tunnel.

Seeing that we have reduced the likelihood that an opponent can get to know the destiny Entity, although we continue to leak information about the possibility that an adversary manages to find the package trace and therefore know the origin Entity .

We propose a method to solve this leakage of information. The method is to set a little padding in all the Entities, and send the file size, so that when an Entity decrypts the packet can get the file size and can add padding equal to the size of file from the destiny Entity appears.

7.1.3 Small padding increased in reception protocol to a size equal to file size only in Entities after destiny Entity appears

For this test we put a padding equal to 1KB, which is relatively small, and add the file size as a parameter, inside the package encrypted.

When receiving the packet, Entity decrypts the packet and if the destiny Entity has appeared, add a padding for the next Entity of a size equal to the file size.

If we look at Figure 10, we see how the processing time of each Entity, in this test with 1MB file and one user in the system is now much more distributed, so the performance is better.

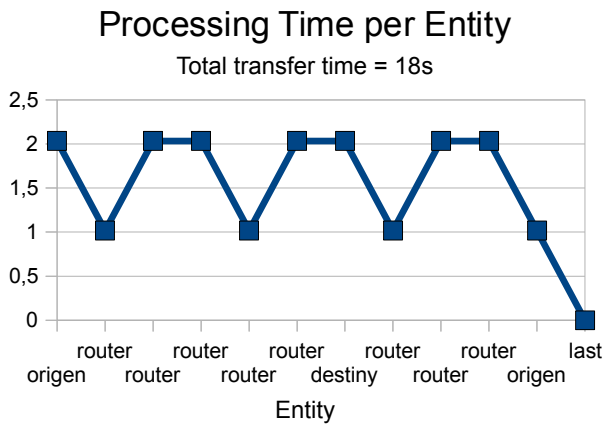

Figure 10. Processing Time per Entity

In addition, the total transmission time was 18 seconds, so we have also improved the latency of the system.

If we analyze the size of packets that arrive at each Entity, would give a result as shown in Figure 11.

As can be seen in this test, an opponent who controlled the entire tunnel of the system could not find anything through the size of packages that arrive in the Entities because they are practically identical in size.

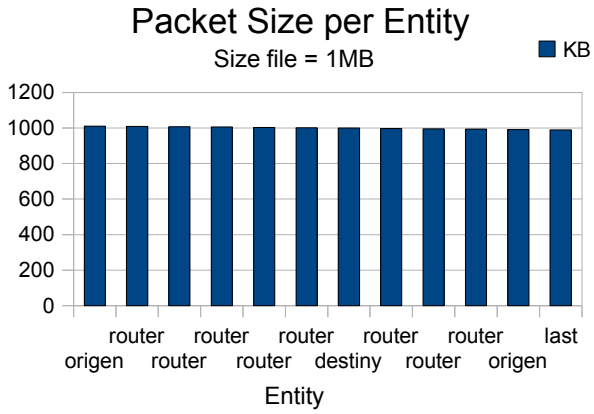

Figure 11. Packet Size per Entity

Therefore, this test does not show the leakage of information about the size of packets that arrive at the Entities, and so, it is increasing the degree of anonymity in the system.

Figure 12 shows the result of variation of the size of packages with the next Entity. Observing the packet size input and output, an adversary can assume that the bytes missed have been used for Entity.

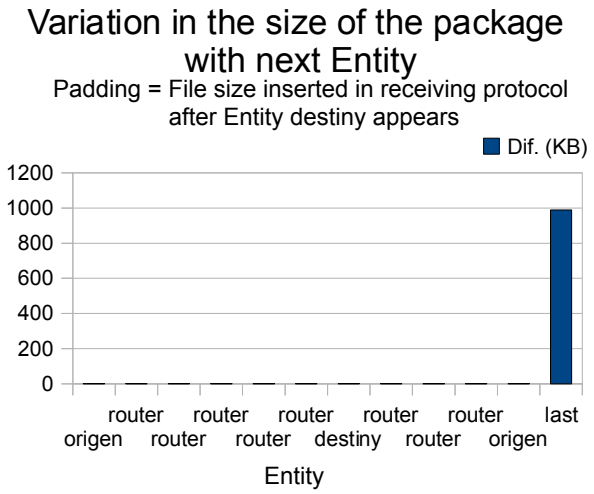

Figure 12. Variation in the size of the package with next Entity

As we can see, the variation in all Entities, except the last, is very small. An adversary could not determine with the variation of the packet size which is the destiny Entity or the origin Entity, can only see what is the Entity that uses more bytes that will be the last Entity of the chain and this is never the destiny or the origin Entity.

Therefore, an opponent might think that the destiny Entity is the last, cause the others looks do not take enough information to be the destiny Entity.

With this setting of the Padding of Protocol provide a better load balance, with a processing time per Entity more shared between Entities of the chain, better performance of system, total time of transfer

decrease and also providing high degree of anonymity, preventing that an adversary that controls the entire tunnel can guess who is the destiny, origin or relay Entity through the packet size or the difference in size of packets that arrive at each of the Entities.

7.2 Timeout test

To calculate the most optimal timeout, we experiment with one user on the system, that is, 12 Entities at least, and calculate the total transfer time and transfer time per KB sent.

In the next Table 1, we can observe the behavior of the protocol for a file size ranging from very small file, 258 KB, until 8 MB file.

As can be seen when we extract the seconds that the system need for each KB, speed in all cases is exactly the same, so use this data to calculate the timeout.

Size File (KB)	Users	Tunnel Lenght	Transfer Time (s)	sec/Entity	Timeout (s)
258	1	12	**5**	0,02	**17,16**
516	1	12	**9**	0,02	**22,32**
778	1	12	**14**	0,02	**27,56**
1024	1	12	**18**	0,02	**32,48**
2048	1	12	**33**	0,02	**52,96**
3072	1	12	**51**	0,02	**73,44**
4096	1	12	**72**	0,02	**93,92**
5120	1	12	**84**	0,02	**114,4**
6144	1	12	**103**	0,02	**134,88**
7168	1	12	**122**	0,02	**155,36**
8192	1	12	**137**	0,02	**175,84**

Table 1. Timeout calculate

The seconds per KB in all cases is 0.02 sec / KB, so we decided to add a timeout greater to:

$$Timeout > Size\ File \cdot 0{,}02\ sec/KB$$

As the timeout must be greater than this multiplication, we propose the following timeout:

$$Timeout = (Size\ File \cdot 0{,}02\ sec/KB) + Tunnel\ length$$

Where we add one second more to the timeout for each entity in the system, as you can see in Figure 13, in all cases, the transfer time is within the threshold of the proposed timeout.

Now we test the timeout but this time changing the number of users in system and leaving a fixed file size to 1MB. In Table 2, we see the variation of users from 1 to 10. This time we added the seconds that needs every Entity according to total transfer time.

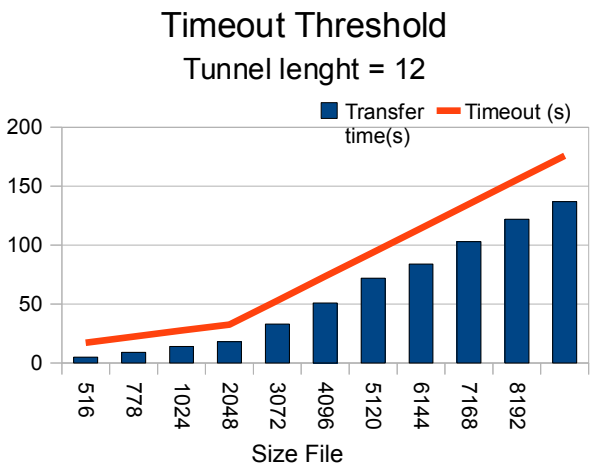

Figure 13. Timeout Threshold

Size File (KB)	Users	Tunnel lenght	Transfer Time (s)	sec/Entity	Timeout (s)
1024	1	12	**18**	1,5	**32,48**
1024	2	15	**24**	1,6	**35,48**
1024	3	16	**25**	1,56	**36,48**
1024	4	18	**29**	1,61	**38,48**
1024	5	20	**32**	1,6	**40,48**
1024	10	30	**47**	1,57	**50,48**

Table 2. Timeout calculate

As is shown in Figure 14, the transfer time ever closer to the threshold timeout, so if users increase may be the time it takes to transfer is greater than the timeout.

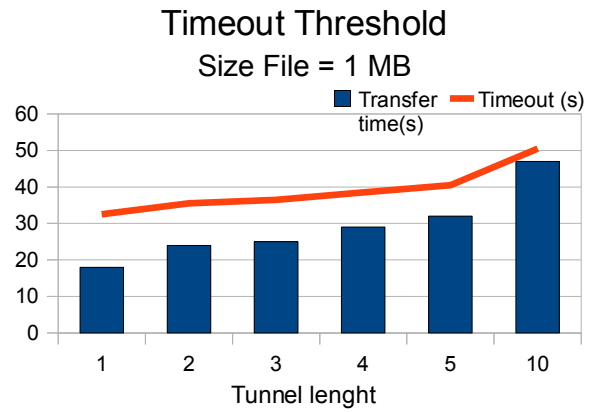

Figure 14. Timeout Threshold

We check it doing tests tests varying between 1 and 50 users, and comparing the transfer time to the timeout.

Table 3 shows the results of the tests, which effectively when users are over 10, the calculated timeout is too small.

In Figure 15, we can observe the evolution of the timeout on the growth of Entities in the system. Therefore, the calculation of the timeout, not only depends on the size of the file, it depends on the number of Entities in the system.

Size File (KB)	Users	Tunnel Lenght	Transfer Time (s)	sec/Entity	Timeout (s)
1024	1	12	**18**	1,5	**32,48**
1024	2	15	**24**	1,6	**35,48**
1024	3	16	**25**	1,56	**36,48**
1024	4	18	**29**	1,61	**38,48**
1024	5	20	**32**	1,6	**40,48**
1024	10	30	**47**	1,57	**50,48**
1024	15	40	**63**	1,58	**60,48**
1024	20	50	**88**	1,76	**70,48**
1024	25	60	**110**	1,83	**80,48**
1024	30	70	**130**	1,86	**90,48**
1024	35	81	**151**	1,86	**101,48**
1024	40	90	**172**	1,91	**110,48**
1024	45	100	**179**	1,79	**120,48**

Table 3. Timeout calculation

Size File (KB)	Users	Tunnel Lenght	Transfer time(s)	sec/Entity	Timeout (s)
1024	1	12	**18**	1,5	**44,48**
1024	2	15	**24**	1,6	**50,48**
1024	3	16	**25**	1,56	**52,48**
1024	4	18	**29**	1,61	**56,48**
1024	5	20	**32**	1,6	**60,48**
1024	10	30	**47**	1,57	**80,48**
1024	15	40	**63**	1,58	**100,48**
1024	20	50	**88**	1,76	**120,48**
1024	25	60	**110**	1,83	**140,48**
1024	30	70	**130**	1,86	**160,48**
1024	35	81	**151**	1,86	**182,48**
1024	40	90	**172**	1,91	**200,48**
1024	45	100	**179**	1,79	**220,48**
1024	50	110	**209**	1,9	**240,48**

Table 4. Timeout calculation

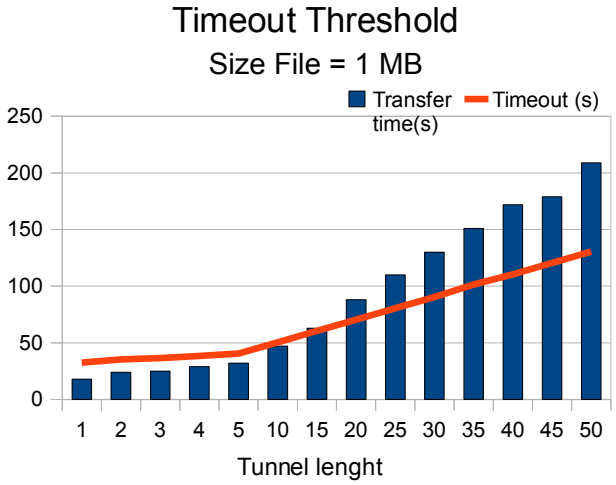

Figure 15. Timeout Threshold

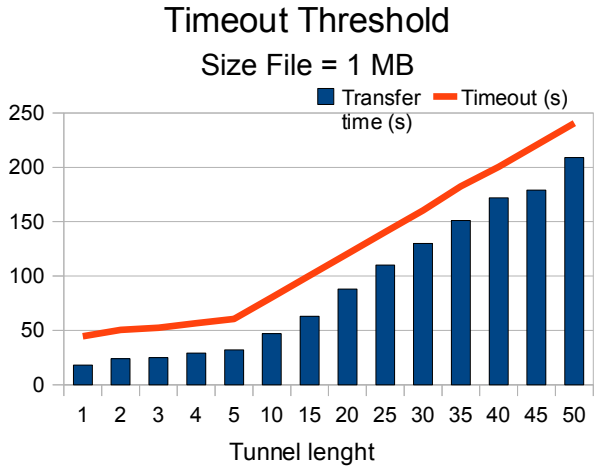

Figure 16. Timeout Threshold

Therefore, we modify the way to calculate the timeout, so it also depends on the number of Entities in system. If you look at Table 3, we see that each Entities need between 1,5 seconds and 1,9 seconds, so we should make sure that never the timeout is less than 2 seconds per Entity in system.

*Timeout = (Size File · 0,02 sec/KB) + (2 * Tunnel lenght)*

If we calculate the timeout for the same tests as in the previous section, we will see if this calculation is valid.

Table 4 shows the test results, and as can be observed there is no transfer overcome the calculated timeout.

Also if you look at Figure 16, we see how not only the transfer time does not exceed the threshold of the timeout, it is also clearly seen that the timeout was increased in proportion to the Entities and also grows in proportion to the size of the file.

Thus we get the most optimal timeout for our protocol.

7.3 Analysis of anonymity

Certificates of Directory, Entities and user signed by a authority, it provide authenticity to the participants of the system, which will be a very difficult for an opponent to get into the circuit in impersonating some of the nodes of the circuit .

Moreover it provide anonymity with the structure, the size of the tunnel and the strategy of our broadcasting system.

7.3.1 System Structure

Assuming that our system has 5 users and therefore the size of the tunnel (N) of 20 hops in the system and as many malicious nodes as M, and without consider other strategies of attack, the entropy can be calculated as:

$$H^2(X^3) = \log_2 N$$

2 H: Entropy
3 X: Anonymous communication system

And the anonymity degree of the system with malicious node M is:

$$D^4(X) = \log_2(N-M) / \log_2 N$$

In Figure 17, we can see how the degree of anonymity decreases as we increase the malicious nodes, as expected, but on the other hand can be seen as the degree of anonymity goes down very slowly, and even when 75% of the circuit nodes are malicious, the level of anonymity is still relatively high. This is because we are in a P2P system.

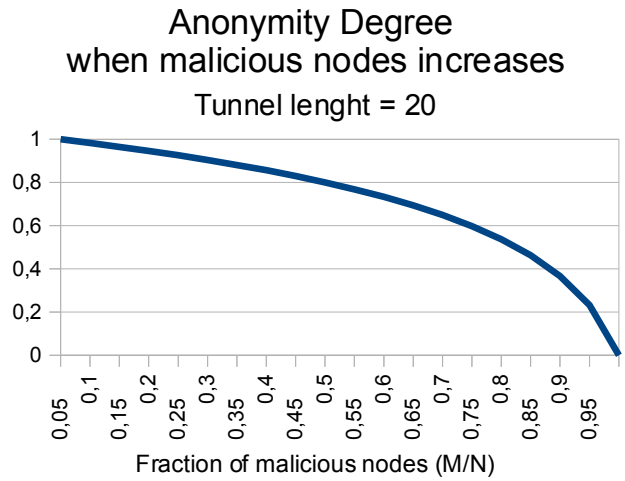

Figure 17. Anonymity degree when malicious nodes increase

7.3.2 Routing Strategy

Figure 18 shows the size of the tunnel variable with users of a network.

As we can see the minimum size of our circuit is ten, which is the size when there are no user and where the degree of anonymity per tunnel size begins to increase more slowly, since against greater is the tunnel size more chances has an opponent to be part of this.

As noted above, in Figure 16, against the greater tunnel size, the transfer time of the document will be higher. This means that against more anonymity we provide through the tunnel size, the latency of the system increase.

An attacker needs to know the IP address of the node to control it [3]. The protocol implemented, an attacker can not know the IP address of the nodes participating in the circuit, because we use SSL protocol, however, if the malicious node is the first, this could have the IP node transmitter.

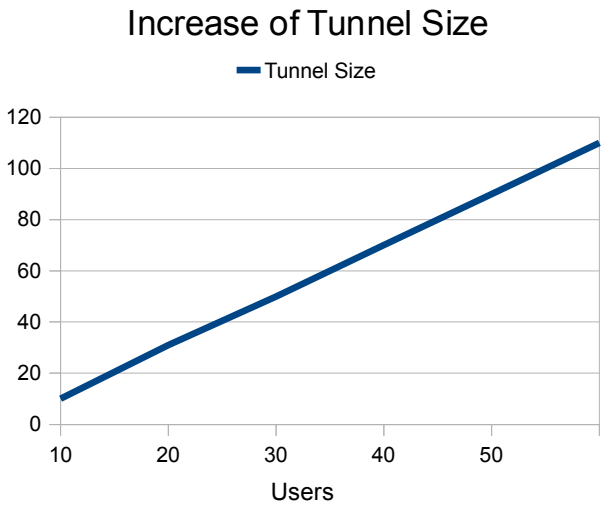

Figure 18. Increase of Tunnel Size

Analysis of the size of the tunnel Andrew Tran, Nicholas Hopper i Yongdae Kim proposed a tunnel size variable to increase the degree of anonymity in the system [3]. Figure 19 shows the level of anonymity in our system, with tunnel size variable.

As the relay strategy is based on onion routing, an enemy has to control the first node and some other nodes in the tunnel, to identify the initiator, and the it need control at least more than half of the nodes in the tunnel for its reconstruction.

The probability that adversary identifies the transmitter node, in our system with variable size tunnel [2]:

$$P(S)=\sum((1-f)^{(l-1)})f(M/N)^{((l+1)/2)}$$

The variable *f* is set by the probability that a relay node retransmit the message of the initiator, which are all nodes before the destiny Entity, since if an adversary controls the nodes that are after the destiny Entity, it can get nothing. Therefore the degree of anonymity in the variable tunnel is:

$$D(X) = (1-P(S)) \log_2(N-M) / \log_2 N$$

As we can see, the tunnel of variable size provide to the system a high degree of anonymity, since even when the fraction of compromised nodes in the system is high, eg 70%, this continues to bring a degree of anonymity in the system .

The value of *f* is 0.25, 0.75 or 0.5 when the destiny Entity is in the first quarter, the middle, or last quarter of the tunnel, respectively.

4 D: Anonymity degree

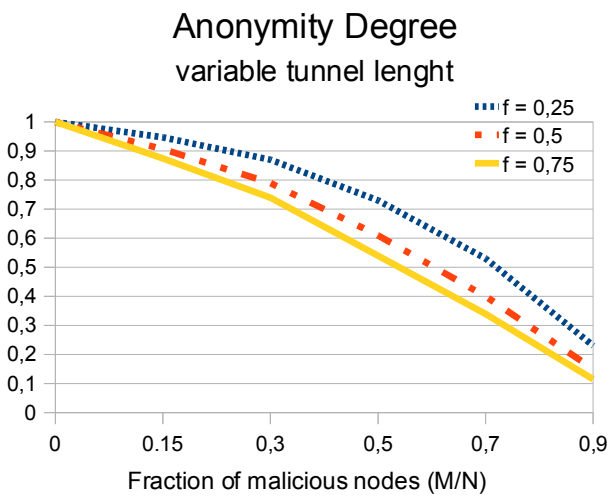

Figure 19. Anonymity degree with variable tunnel lenght

The authors Jia Zhang, Haixin Duan, Wu Liu i Jianping Wu shows an analysis of tunnel size between the variable and fixed, which show that the tunnels with a variable size provides a higher degree of anonymity that the tunnels of a fixed size.

8 COMPARATION WITH OTHER CURRENT SYSTEMS

Our protocol performs iterative searches mode, as is done in other systems such as SALSA, and also in this system can only be broken if an opponent is the first node.

In SALSA, are also customers who choose the construction of the circuit, although each node only meets the predecessor and successor, exactly like our protocol.

However, the system only selects random 3 nodes to build a circuit, and in our case the minimum number of nodes in a circuit is 12, and is variable as it increases the number of users in the system.

Our algorithm encrypts the symmetric key in an encrypted public key of the node, while in other systems such as MorphMix a node establishes a symmetric key by sending a node b half of the algorithm encryption with Diffie-Hellman public key witness w. The node b transmits encryption to W, it removes the layer of encryption and sends the result to a. This generates half of the Diffie-Hellman algorithm and sends through b. Thus, both parties can generate the shared symmetric key.

The Cashmere system also uses virtual re-transmitter, although made up by a set of nodes for the resistance, has groups where each transmitter has a public key and private individuals and other public and private keys for the group.

In that form the system uses the Group ID as a key to send messages. The root of group is responsible for processing messages on behalf of the re-transmitter group.

As we see there are other systems that uses features similar to ours, but with our algorithm we managed to raise the level of anonymity, since we do not just use a circuit with a variable number of nodes and 10, which is the strip where the degree of anonymity is achieved, by a circuit node begins to stagnate, but in addition we have managed to obtain more anonymity on the possibility that an adversary that measure packages are sent between nodes, as it will be very difficult to find out who is the source or destination node, observing the size of the packages, to have all the same size and all seems to take virtually no info to get padding after reading the package.

If we look at the table we can see other systems that use the same strategies for both the structure and size of the tunnel or the retransmission strategy.

Strategy	Anonymity	Performance	Other systems
Unstructured	High	-	MorphMix
Variable length	High	Low	AP3, Crowds
Onion routing	High	Mid	Tarzan, MorphMix

Table 5. Strategies implemented protocol

According to the article [3], we can see how is done a comparison of different systems, although we see that none of the systems implemented provides a high anonymity as our protocol both in structure and in the size of the tunnel, such as in as retransmission strategy.

And finally, we have managed to obtain an optimal distribution of load among all the Entities, which gives better performance in the protocol.

9 CONCLUSIONS

We have achieved a reliable protocol, because the User Application knows the mistakes that occur during the circuit and it could try to create a new one.

Improved efficiency and anonymity to multiplex TCP flows along with each circuit.

The User Application can send traffic to any node of the circuit, is random, and the destination can never be the last node, in this way decrease the chances that an adversary can perform attacks based on the final point of the circuit.

Our protocol provides a directory server signed by CA server and provide certificates for each of the Entities and users.

By using a symmetric key data encryption, and by encrypting this with the public key of the node, and sending it along a circuit of nodes chained and encrypted consecutive, we have privacy into the system.

We got to raise the degree of anonymity with a method in the process of receiving where Entities set a padding of the same size as the file we are sending to all Entities that follow to destiny.

When the degree of anonymity increase on the system cause of the size of the circuits, the latency increases.

Therefore, the protocol provide us a high degree of anonymity because will be very difficult for an opponent to guess who is the transmitter or who is the destiny, due to the method of use of padding and variable size of the tunnel.

It has a built-in timeout optimal, that is, a timeout proportional to the file size to send and the number of entity that holds the system.

Distribute the load among all Entities, because it is better to have more services that require fewer resources than have less services but they need more resources. This has improved the performance of the protocol.

If the first node of the tunnel is malicious, it could identify the actual user, but not really know if the sender or just a relay, but may not know the real server. If however, the first node is honest, another node of the tunnel can not identify the sender directly and there is only one real chance 1 / N (N number of nodes in the tunnel) to determine the correct identity of the sender.

The initiator sends traffic to the destiny node of the circuit, which is never the last, thus is avoiding attacks based on the observation of the end of the circuit.

It is very difficult to determine whether the outcome of the request is correct [3]. In the protocol has been implemented an algorithm where the user will get a response of the transfer, as if it is correct as if it has failed. In section 7.2, we have been calculating the timeout to give a respond as if an error has appeared in transfer as if the package has been lost along the way.

10 FUTURE WORKS

Algorithm created in localhost, therefore they could do tests by implementing it within protocol in a decentralized network system such as CoDeS, system provided by the UOC University, which allow services to deploy on the computers voluntarily contributed to a community.

It could optimize the operations of the protocol, when we create the circuit, if we put a random parameter between 0 and 1, all Entities that precede the Entity destination, so that only some of them appear in the circuit.

It can give meaning to the algorithm if we implement it in a context where there are a teacher and student roles, each having different permissions. In addition, we can implement in protocol the method of obtaining or modifying a document server. So we could get a complete decentralized application to do peer-reviews anonymously.

11 REFERENCES

[1] Edman, Matthew and Yener, Buelent: On Anonymity in an Electronic Society: A Survey of Anonymous Communication Systems. Review. NY-USA, 2009.

[2] Jia Zhang, Haixin Duan, Wu Liu and Jianping Wu: Anonymity analysis of P2P anonymous communication systems. Article. Tsinghua University, 2011.

[3] Andrew Tran, Nicholas Hopper and Yongdae Kim: Hashing it Out in Public. Article. Chicago-USA, 2009

[4] Roger Dingledine, Nick Mathewson and Paul Syverson: The Second-Generation Onion Router. Article. San Diego-CA, 2004.

[5] Qiyan Wang, Prateek Mittal and Nikita Borisov: In Search of an Anonymous and Secure Lookup. Article. Chicago-USA, 2010

[6] Michael J. Freedman, Emil Sit, Josh Cates and Robert Morris: Introducing Tarzan, a Peer-to-Peer Anonymizing Network Layer. Berlin-Germany, 2002.