Towards a Secure Data Sharing Peer-to-Peer Network based on Geometric and Semantic Distances

Ana-Delia Sambotin, Mugurel Ionut Andreica

TOWARDS A SECURE DATA SHARING PEER-TO-PEER NETWORK BASED ON GEOMETRIC AND SEMANTIC DISTANCES

Ana-Delia Sâmbotin, Mugurel Ionuţ Andreica
Department of Computer Science
Politehnica University of Bucharest
Splaiul Independenţei 313, sector 6, 060042, Bucharest
Romania
E-mail: delia.sambotin@gmail.com, mugurel.andreica@cs.pub.ro

KEYWORDS
P2P, virtual geometric coordinates, semantic distance, security, multicast, data sharing.

ABSTRACT

In this paper we propose a new strategy that can be applied for creating a secure peer-to-peer topology in which the identity of the source node cannot be revealed. The main goal is to obtain a decentralized network distributed in space, where the users are allowed to share and exchange their music files. The proposed model uses different metrics for estimating the distance between nodes (like the round trip time and the semantic distance) and uses the smallest values in order to select a node's neighbors. For the identity protection, the system imposes the encryption of the traffic and that the communication is mediated by a node from the network, randomly chosen by each instance.

INTRODUCTION

Peer-to-peer is a very popular technology which connects thousands of clients in a decentralized environment. They are used in a large number of situations, from the simple need to transfer a file, to more complex interactions like social networks or resource sharing. P2P is a popular technology especially used for file sharing because it allows the client applications to upload and download files over the network, without the need for some special server devices. This means that the peers from the network are both suppliers and consumers of resources, in contrast to the traditional client-server model where the suppliers are only the servers, and the clients are the consumers. There are also some disadvantages in peer-to-peer architectures. An important drawback is that P2P networks are typically less secure than a client-server network because security is handled by the individual computers, not by the network as a whole. The resources of the computers in the network can become overburdened as they have to support not only the workstation user, but also the requests from network users.
The main concepts behind the peer-to-peer networks are: sharing resources, decentralization and self organization. There are several types of peer to peer networks, based on the centralization degree of the overlay. In a fully decentralized network there is no intermediary device which keeps track of the peers position and activity. This type of network is very scalable because the failure of a peer does not affect the entire network. The hybrid architecture consists of a central server which keeps the information about the users from the network as metadata. This kind of architecture can be classified in centralized indexing (the peers maintain active connections with the server) and decentralized (the server maintains connections with a set of peers, named supernodes). In this case we denote the server to be the component delegated with the role of supplying the basic information essential for a peer to connect the overlay. Due to the lack of a server device the peers must be able to organize themselves in order to obtain a stable network. There are different ways that the peers could connect with each other depending on which properties the topology wants to improve.
IP multicast provides a method of efficient many-to-many communication in contrast to the one-to-one model of IP unicast, in which data packets are sent from a single source to a single recipient. This concept is becoming increasingly important, both in the Internet and in private networks. The multicast technology enables the use of the following applications: video conferences, live broadcasting, web TV, web radio, video-on-demand, e-learning, whiteboard data exchange.
There are several ways to simulate a multicast network. The most intuitive and efficient one is to construct a network infrastructure, where the intermediary devices duplicate the received packets and send it to a group of users. This approach is not quite efficient due to the fact that the service providers must change the structure of the network and they must buy new equipments. These changes can be expensive and take a long period of time. Another way to simulate the multicast communication is to build an application layer multicast overlay; thus, the application is responsible with the multiplication of the received data and with forwarding it to other nodes. This solution is less expensive but it uses more of the client's resources.
In this paper we propose several methods in order to construct a secure data sharing peer-to-peer network with multicast capabilities. The work presented in this paper is a continuation of the one presented in (Sâmbotin and Andreica, 2011).

RELATED WORK

In this section we present other approaches for some similar applications. We were interested in other similar approaches of building secure peer-to-peer networks and multicast

overlays.

Peer-to-Peer Spatial Cloaking Algorithm

A location-based service (LBS) is an information service that needs to know the positioning of a user (mobile device) in order to provide some useful information, like where is the nearest bank. In (Chow et al., 2006) the authors propose a peer-to-peer spatial cloaking algorithm for mobile devices, which will help the users to protect their private information without seeking help from any centralized third party. A practical example could be when a mobile user looks after the nearest gas-station. For security reasons, the user will first find other peers to collaborate as a group and will cloak its exact location into a spatial region that covers the entire group. The next step assumes the random selection of one node within the group which will be delegated with the role of an agent. This means that the communication with the location-based database server will be mediated by the agent. Because the server processes the query based on the cloaked spatial region, it can only give a list of candidate answers that includes the actual answers and some false positives. After the agent receives the possible answers, it forwards it to the mobile user that requested it.
The described model can function in two modes: on-demand or proactive. The first one assumes that a mobile device will start this process when they need a information provided by the location-based service. In the second model, the users periodically search for different devices in order to form a group that will hide his location.

CAN

In (Ratnasamy et al., 2001) the authors proposed a new strategy in which the network is composed of many individual nodes and in which the space is split between them. The CAN network resemble a hash table and the basic operations that can be performed in this model are: the insertion, lookup and deletion of *(key,value)* pairs. Each CAN node stores a chunk (called a zone) of the entire hash table. In addition, a node holds information about a small number of "adjacent" zones in the table. Requests (insert, lookup, or delete) for a particular key are routed by intermediate CAN nodes towards the CAN node whose zone contains that key. The CAN design is completely distributed (it requires no form of centralized control, coordination or configuration), scalable (nodes maintain only a small amount of control state that is independent of the number of nodes in the system), and fault-tolerant (nodes can route around failures).
The entire CAN space is divided amongst the nodes that coexist in the system. When a new node joins, the system must be allocated its own portion of the coordinate space. This is done by an existing node splitting its allocated zone in half, retaining half and handing the other half to the new node. A major problem that a peer encounters is finding the proper zone. In order to achieve its purpose the new node then randomly chooses a point *P* in the space and sends a *JOIN* request destined for point *P*. This message is sent through the network via any existing CAN node and by using the routing mechanism.
Under normal conditions each node sends periodic update messages to each one of its neighbors giving its zone coordinates and a list of its neighbors and their zone coordinates. When a prolonged absence of an update message from a neighbor is noticed then it is interpreted as a failure. Once a node has decided that its neighbor has died it initiates the takeover mechanism and starts a takeover timer.

Multicast Overlays over Peer-to-Peer Networks

As we can find in (Tan and Jarvis, 2007) there are different ways to build a multicast overlay depending on the properties that we want to improve. This paper describes and analyzes several ALM protocols which aim to construct a topology for multicast communication based on the most significant properties (application domain, group configuration, routing protocols). Also, the authors compare the IP multicast with the different ways of implementing a multicast network at the application layer, highlighting the advantages and disadvantages for each technology.
The concept of ALM (Application Layer Multicasting) concerns the implementation of multicasting functionality as an application service instead of a network service. This solution was considered due to the fact that one-to-many or many-to-many communication should use efficiently the network resources. A multicast network service implies that the local network must be equipped with routers that are capable of setting up and tearing down IP Multicast sessions as well as processing and routing IP Multicast packets. Therefore, instead of relying on the local internet providers to enable the multicast communication, we can assign the duty of creating a multicast overlay to the application layer. While IP Multicast is implemented by network equipment (routers) and avoids multiple copies of the same packet on the same link as well as possibly constructing optimal trees, ALM is implemented by the application and can lead to multiple copies of the same packet on the same link as well as typically building non-optimal trees.
ALM's disadvantages, such as longer delays and less efficient network usage compared to IP multicasting, are balanced by its advantages such as immediate deployability on the Internet, easier maintenance and update of the algorithm, and, last, but certainly not least, the ability to adapt to a specific application.
Some of the most popular ALM protocols that we will discuss are:

- ZIGZAG (Tran et al., 2004)
- NICE (Tran et al., 2004)
- OMNI (Banerjee et al., 2003)

ZIGZAG is a single source, degree-bounded application layer multicasting approach for media streaming. It arranges receivers into a hierarchy of clusters and builds a multicast tree on top of it. After applying the recursive rules of organizing the nodes into a multi-layer hierarchy of clusters, it will result that the nodes closer to the root will have large out-degrees and will run out of their bandwidth quickly. This effect might not be acceptable for bandwidth-intensive media streaming applications. This protocol provides a mechanism for maintaining a connected tree: when the parent node leaves the network without an announcement, the delegated node will help the children to reconnect immediately to the new parent. ZIGZAG periodically runs optimization algorithms to improve the quality of service.
NICE is an acronym which stands for the NICE Internet

Cooperative Environment. This scalable application layer multicast protocol uses a hierarchical clustering approach to support a larger number of receivers. NICE was designed to provide a topology for low bandwidth soft real-time data stream applications such as real-time stock quotes and updates and Internet radio. This approach detects inaccurate placement of hosts in clusters on different layers and gradually moves to a global optimal hierarchy. NICE allows nodes in a cluster to exchange periodic messages in order to maintain the appropriate peer relationships. In every cluster it will be a delegated node that will be responsible with maintaining the proper size of the structure and with identifying the departure of a peer in order to keep a connected tree.

OMNI stands for the Overlay Multicast Network Infrastructure and offers a multicast overlay for an efficient transfer of media streams. This method will generate a topology that consists of a set of devices called Multicast Service Nodes (MSN). These points are distributed in the network and provide efficient data distribution services to a set of peers. A client will be associated with a single MSN to receive multicast data service. The MSNs themselves run a distributed protocol to organize themselves into an overlay which forms the multicast data delivery backbone. The data delivery path from the MSN to its clients is independent of the data delivery path used in the overlay, and can be built using a network layer multicast or an application-layer multicast system.

Traffic Analysis

Traffic analysis is the process of intercepting and examining messages in order to deduce information from patterns in communication. It can be performed even when the messages are encrypted and cannot be decrypted. Thus, for hiding the content of a message we can use encryption with public/private/symmetric keys, whereas traffic padding may be used to hide the traffic pattern.

In (Jiang et al., 2001) a strategy for improving the security in a wireless network is proposed. The authors developed a suitable cover mode with the objective of minimizing the energy consumption because usually, the peers from these types of networks are mobile devices with limited power supply. They also want to minimize the quantity of the dummy traffic because these information incurs an overhead.

In order to prevent traffic analysis, the authors considered that it is important to hide not only the real traffic pattern, but also the changes in the real traffic pattern.

A cover mode is considered to be constructed so an intruder cannot determine the real operation mode of the network at any given time. The paper proposes several methods for building a cover mode, like: end-to-end and link. An end-to-end cover mode maintains a constant rate of traffic between each (source, destination) pair, independent of the actual operation mode while a link cover mode is obtained by achieving constant traffic rate on each link in the ad hoc network, independent of the actual operation mode. Unlike the end-to-end cover mode, link cover mode is implemented by inserting dummy packets on each link, so as to maintain a constant rate on that link.

The results that the authors obtained indicated that end-to-end cover mode generally performs worse than link cover mode, but in large networks, the two approaches yield similar energy overheads.

SYSTEM DESIGN

This section presents the application's purpose and its architecture. Furthermore we present a detailed description of each main component within the application and how these components are connected together. This section also presents each stage and the communication protocol used between the involved entities.

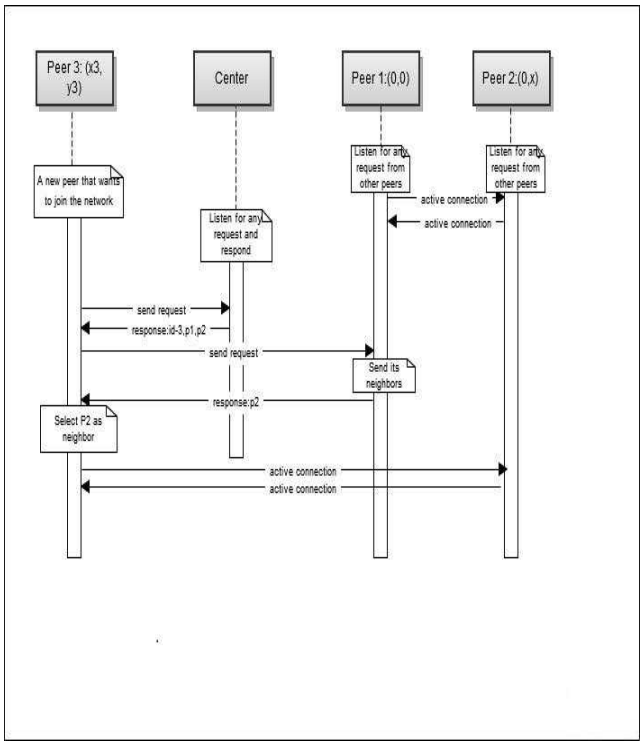

Figure 1: Multicast overlay construction steps.

Application's Purpose

The main goal of our system is to create a new strategy for the constructions of a secure decentralized data sharing peer-to-peer overlay. This means that there isn't any entity which knows the entire network, but only fragments, and by fragments we understand the neighbors and the extended neighbors. Also, the system has the responsibility to protect the user's identity and to cover the traffic within the network.

This model aims to enhance the security of the network where the identity of the source transfer cannot be identified. In order to achieve this, a new role of a mediator will be defined for a regular peer. Also, all the traffic within the system will by encrypted using symmetric keys.

This network must be completely decentralized in order to improve the stability issue (the failure of a node must affect a minimum number of peers). The final representation of the overlay should be as a tree structure, where the nodes have a reasonable number of neighbors; thus, the topology will not be considered to contain peers with the role of a server.

The application must allow a multicast communication over the peer-to-peer overlay. Thus, we want to construct an application layer multicast which will allow one-to-many communications. Each node should be able to send the same

message to any of its neighbors, regardless of the requesters, and to receive messages.
Another aim of this system is that its components should have a low level of dependency. Thus, the development of a new strategy should be easily integrated with the rest of the system.

The Construction of the Overlay

In Fig. 1 we can identify the main stages a peer must go through in order to join the network.
Before starting the process of obtaining its coordinates, a peer must share a directory with music files, which is analyzed when the application starts. Thus, we are interested in extracting some additional information (like title, artist, album, genre) from these files and form a vector of tokens. Each token (an element from the vector) will have a weight value associated to it, which is computed based on metrics like term frequency and inverse document frequency. In other words, the weight assigned to a term using these metrics should be:

- highest when the term occurs many times within a small number of documents (thus lending high discriminating power to those documents);
- lower when the term occurs fewer times in a document, or occurs in many documents (thus offering a less pronounced relevance signal);
- lowest when the term occurs in virtually all documents.

The first step is to contact the bootstrap node in order to obtain the minimum amount of information needed for joining the network. It receives the data about the last three nodes that joined the system.
Furthermore the peer must contact each node in order to update the data associated with each peer. Mainly, we are interested in obtaining the coordinates of those nodes. Knowing the identity of its neighbors, the peer will ping each one of them and will compute the obtained RTT value. This value is combined with the semantic distance computed between the two nodes. The similarity (semantic distance) between two peers is computed using the cosine similarity that measures the cosine of the angle between the token vectors of the two peers. The result can take a value from 0 to 1, where 0 means that the two nodes have nothing in common, and 1 meaning that they have shared identically the files. We consider that the obtained value represents the distance between this two nodes (Skvortsov and Kostyuk, 2006) .

$$\text{Sim}(A, B) = \cos\theta = \frac{A \bullet B}{|A||B|} = \frac{x1*x2 + y1*y2}{(x1^2 + y1^2)^{1/2} \ (x2^2 + y2^2)^{1/2}}$$

Figure 2: Semantic distance formula.

Knowing some neighbors and the distance to them, the peer will try to compute its coordinates using the following assumptions and formulas. If the peer doesn't receive any data from the bootstrap node, this means that it can be considered to have the coordinates (0, 0). If the peer has only one neighbors with the coodinates (x_n, y_n) then it will choose the coordinates (x_n+dist, y_n). If the node has 2 or more neighbors, then in order to obtain the coordinates of the point we will use the rules for computing distances proposed in (Sâmbotin et al., 2011).

Security Module

In order to enhance the security we developed a different strategy adopted by the nodes from the system that will protect and disguise the identity of a node. The reason we need to do this is based on the issue of rights over the files that a user shares and it is undesirable that the application could provide a mechanism for tracing them.

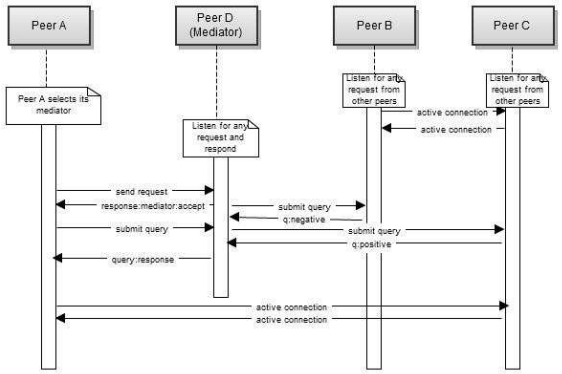

Figure 3: Transfer protocol

Fig. 3 presents the protocol used by the peers when they search for a file. First, we assume that each node shares some files in mp3 format and wants to search and find other songs. In order to retrieve the desired file, the user must submit a query through the network, and whenever a peer receives such a request and considers that it could respond affirmatively, it contacts the initiator. This approach is the most common one. Over this model we introduce the role of a mediator and some rules that will be described further.
Before submitting a query through the network, the peer randomly selects one of its neighbours that will be delegated with the communication between the initiator and the node that will respond affirmatively to the query. Thus, we can consider that the peer cloaks behind its neighbours and the diameter of the group that hides the initiator can increase.
The second step is to implement a mechanism that will protect the system from traffic analysis. For this, each node that sends a query will first establish a set of terms that will be inserted in the initial request. The tokens that are additionally inserted in the request are generated using a specific pattern. This means that they are formed from a randomly generated string that only contains symbolic characters that are appended at the beginning and at the ending of the file. Another used strategy will be that that all nodes will periodically exchange messages (some will be dummy messages and others will have the purpose of maintaining the network and the connections between nodes), independent of the operation mode.
The message exchange assumes that all the traffic is encrypted using symmetric keys. When a mediator receives a list of tokens (some valid and others dummy), it will forward the request to its neighbours.
We must introduce some constraint for the role of a mediator peer. First, when a mediator is chosen by a node, it doesn't have the possibility to deny a request, but he must do everything it can in order to resolve a query. For security concerns, a node with this role will not try to resolve any of

the requests. The only action that it is allowed to do is to forward the received queries. The neighbours will try to resolve the query, and if neither of them will be able to respond affirmatively, then the mediator will become an initiator and will retain the node for which it has this role. This approach is used until a user gets a response or no one can provide the requested file. When a node will respond positively, the system will be able to transfer the data along the shortest path.

Application Architecture

The system is made from the following main modules which interact with one another. Basically, each module represents a main functionality of the system. As we can see in Fig. 4, the main modules are:

- Network components
- Security module
- Listen thread
- Settle Neighbors thread
- GUI
- Input module
- Timeout thread

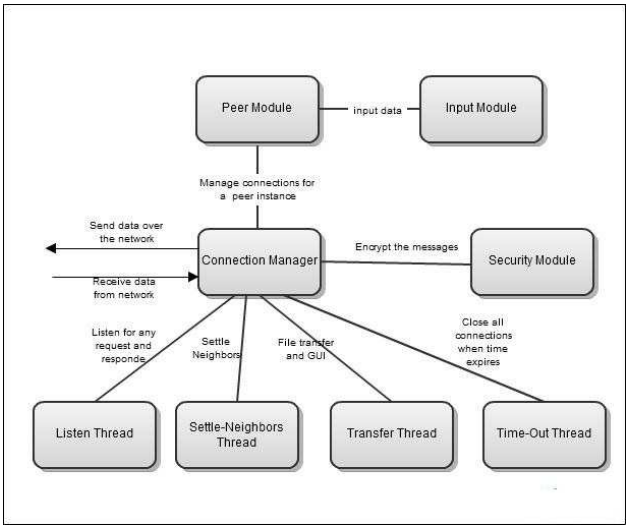

Figure 4: System architecture.

We have chosen this model because we wanted a loose coupling between the main parts. Thus, each component can be easily replaced with another component in the future.

Module Description

The *network module* (connection manager in Fig. 4) is responsible for constructing the overlay that must provide the ability of finding the most appropriate peers for the file transfer. The final goal of the network module is to build a peer-to-peer overlay that can be represented as a tree structure and which will allow the multicast communication.
This module is highly connected to the module responsible with the security within the system. That is because all the traffic is generated according to the strategy described in the previous section, i.e. encrypted using symmetric keys.
One major aspect of the network is that it will be fully decentralized. This means that no peer will have a map with the entire overlay, but it will be aware only of the direct neighbors (with which it will keep an active connection) and its extended neighbors (peers that are the neighbors' neighbors).
The network will be composed of a center node (or bootstrap node), which will not be a part of the network, but will have the responsibility to introduce other peers in the overlay by supplying information about other nodes, and at least one regular peer. In order to keep a consistent set of nodes, the central entity will be contacted before each valid departure of a peer from the network.
The input module is responsible for parsing and gathering the information from the configuration file which contains all the required data that a peer needs in order to start.

```
<config>
        <title>Global Parameters</title>
        <params>
                <ip>192.168.0.121</ip>
                <port>30001</port>
                <bandwidth>150</bandwidth>
                <ipBootstrap>192.168.0.6</ipBootstrap>
                <portBootstrap>30000</portBootstrap>
                <time>30</time>
        </params>
</config>
```

Figure 5: Sample configuration file.

As we can see in the figure above, a peer needs to know its ip, port, the address of the boostrap node. The only parameter that must be changed is the ip parameter with the value.
The *Listen Thread module* is responsible with listening for any request from the network. These may come from another peer and have the following form: "action-info_peer", where *info_peer* is a representation of information regarding the requesting peer and *action* can be:

- *collect* – gather the updated coordinates of the neighbors
- *responseCollect* – the response of a peer with its coordinates
- *remove* – removes a neighbor from its list of neighbors
- *add* – add a new neighbor
- *extended* – ask for the neighbors
- *sendExtended* – send the list of neighbors

The *Graphical user interface (GUI)* is implemented as a different thread in order to organize the resources better. From the GUI window we can visualize some valuable information like which peers are the neighbors and the identity of the current peer.

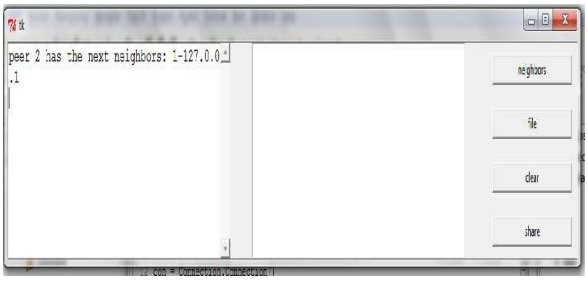

Figure 6: GUI.

The *Time out Thread* has the responsibility of stopping and exiting the program after a period defined by the user. The "time" tag from the xml configuration file is associated with this value.
The application is developed in Python and we only used one external module, Tkinter for the graphical interface. We have chosen Python because it is a powerful, pure object-oriented programming language with efficient data structures.

EXPERIMENTAL EVALUATION

We simulated the construction of the network on one local machine for 50 to 300 peers. For the simulation phase we used a different module which uses the same formulas, strategy and metrics as the ones described in this paper.
In order to test our solution we first randomly generated the coordinates of a point and a number from 1 to 6 (the suffix of the shared folder). When a new peer joins and asks for the distance to its neighbours, it gets the geographical distance between one of these generated points and its neighbors. The value will be processed and combined with the similarity coefficient. We consider that we know the positioning of its neighbors.
We wanted to evaluate if the points will be distributed in the entire space or if they will gather in one area. In order to do so, we tested the system using different weights.
When the network considered only the geographical distances, the nodes were distributed through the entire space (see Fig. 7).

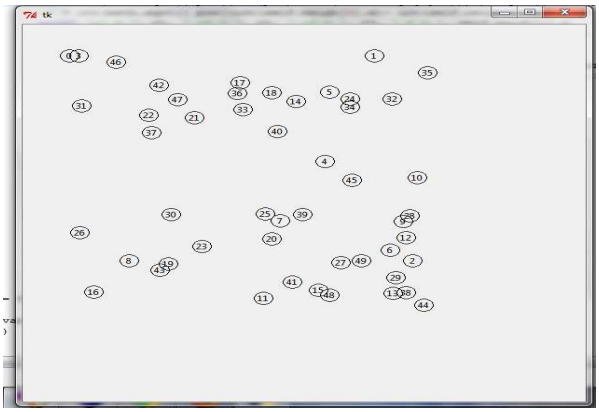

Figure 7: Points distribution (only geographical).

When we used only the semantic distance, the points gathered in one area, because the folders contained approximately the same files (see Fig. 8).

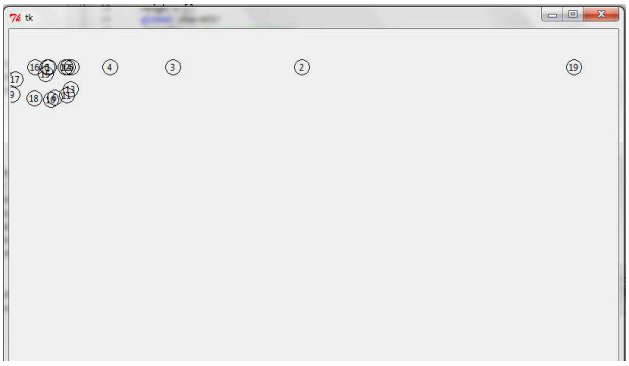

Figure 8: Points distribution (only semantic).

When we attributed an equal percent to each metric, the points formed a distributed group (see Fig. 9).

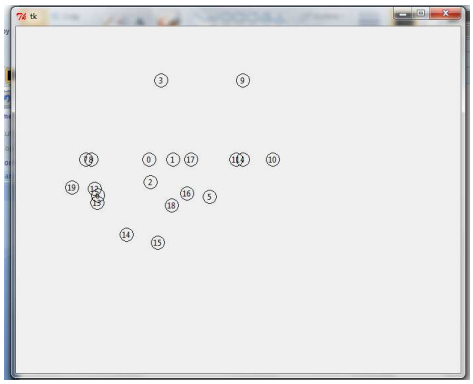

Figure 9: Points distribution (both metrics).

We also wanted to evaluate the security module in order to appreciate how the system will perform. Thus, we measured the time needed for the search operation with and without the mediator role.

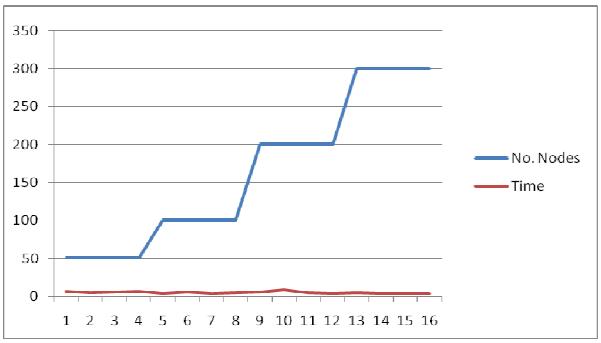

Figure 10: Results for the secure case: with mediator.

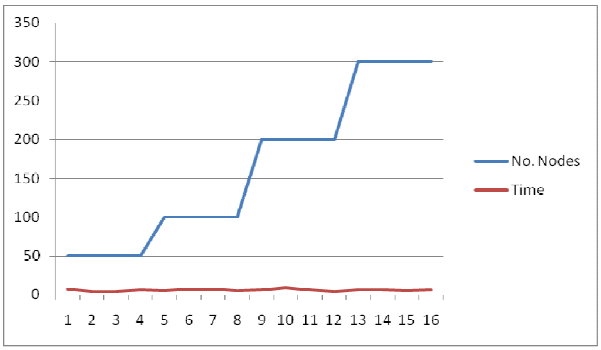

Figure 11: Results for the secure case: without mediator.

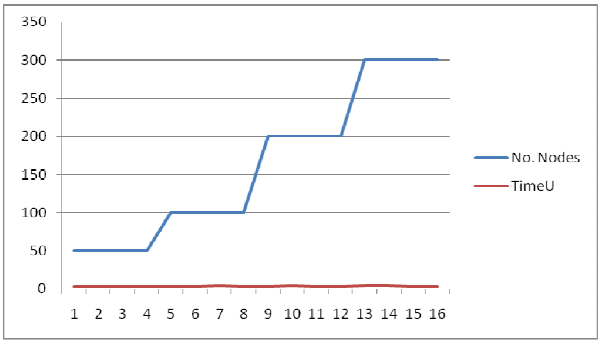

Figure 12: Results for the unsecure case.

As we can observe, the overhead is not induced by the new role of the mediator. We have been expected this because the neighbours are selected according to the smallest distance, which considers the similarity between two nodes. Thus, it is reasonable to affirm that in most cases, the searched file can be located in the proximity of a peer (one of the mediator's neighbors). The most expensive part is the generation of the keys and the exchanging of this entities among the nodes and the process of adding dummy data to the original request.
The security module slows the response time of a node with approximately 35%.
We have also tested the main application. From these test cases we can declare that the communication through the network works properly.

CONCLUSIONS AND FUTURE WORK

As we can observe from the previous section, we obtained a decentralized network which uses two distance metrics in the building phase: a geometric distance over virtual coordinates and a semantic distance. The neighbors of a peer are the closest nodes of that peer. The best scenario is to use both metrics: the structure of the network is more compact but not too congested.
The "cost paid" for a secure network when the content cannot be found is not very large and it is worth "paying". The process that takes longest is the key generation, but this will be performed only in the first phase.
As future work we want to test the application on several different machines. The security tests must consider the latency of the message exchange through the network. The result will be influenced by the quantity of the dummy data added to a request and by the additional number of messages that will be sent between a node and its mediator. Beside the network structure analysis, we are also interested to test the transfer speed of a file from a peer to another.
We also built the application in such a manner that we can easily add other metrics for the distance between two nodes. We want to consider the load of the network on a longer period of time when we compute the distance.

ACKNOWLEDGEMENT

The work presented in this paper has been partially supported by the Sectoral Operational Programme Human Resources Development 2007-2013 of the Romanian Ministry of Labour, Family and Social Protection through the Financial Agreement POSDRU/89/1.5/S/62557, and by CNCS-UEFISCDI under research grant PD_240/2010 (contract no. 33/28.07.2010), PN II – RU program.

REFERENCES

Banerjee, S.; C. Kommareddy; K. Kar; B. Bhattacharjee; and S. Khuller. 2003. "Construction of an Efficient Overlay Multicast Infrastructure for Real-time Applications". In *Proceedings of IEEE INFOCOM*, vol. 2, 1521-1531.

Chow C. Y.; M. F. Mokbel; and X. Liu. 2006. "A Peer-to-Peer Spatial Cloaking Algorithm for Anonymous Location-based Service". In *Proceedings of the 14th Annual ACM International Symposium on Advances in Geographic Information Systems,* 171-178.

Jiang S.; N. H. Vaidya; and W. Zhao. 2001. "Power-Aware Traffic Cover Mode to Prevent Traffic Analysis in Wireless Ad Hoc Networks". In *Proceedings of IEEE INFOCOM*.

Ratnasamy S.; P. Francis; M. Handley; R. Karp; and S. Shenker. 2001. "A Scalable Content-Addressable Network". In *Proceedings of the ACM SIGCOMM Conference*, 161-172.

Sâmbotin, A.-D.; M. I. Andreica; and E. Mocanu. 2011. "Strategies for Assigning Virtual Geometric Node Coordinates in Peer-to-Peer Overlays". In *Proceedings of the 6th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC),* 253-258.

Sâmbotin, A.-D.; and M. I. Andreica. 2011. "Towards a Peer-to-Peer Recommender System Based on Collaborative Filtering Techniques". In *Proceedings of the 18th International Conference on Control Systems and Computer Science (CSCS)*, vol. 2, 537-544.

Skvortsov, A. V.; and Y. L. Kostyuk. 2002. "Compression of Coordinates of Triangulation Nodes". In *Russian Physics Journal*, vol. 45 (5), 472-476.

Tan, G.; and S. A. Jarvis. 2007. "Improving the Fault Resilience of Overlay Multicast for Media Streaming". In *IEEE Transactions on Parallel and Distributed Systems*, vol. 18 (6), 721-734.

Tran, D. A.; K. A. Hua; and T. T. Do. 2004. "A Peer-to-Peer Architecture for Media Streaming". In *IEEE J. Sel. Areas Commun.*, vol. 22, no. 1, 121- 133.