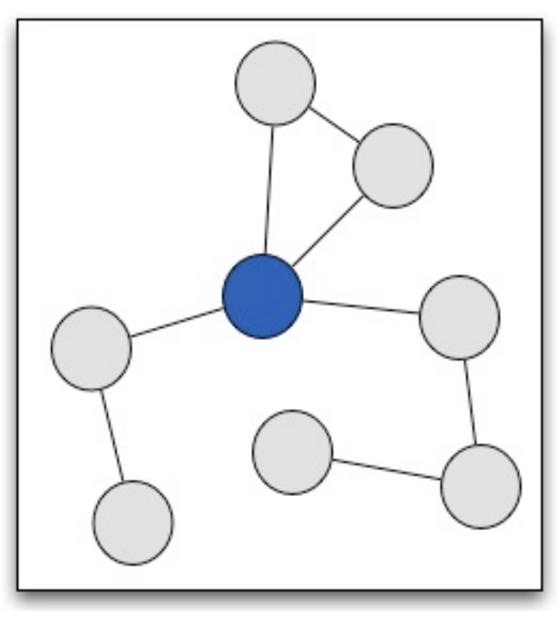
TAG-BASED COOPERATION IN P2P NETWORKS

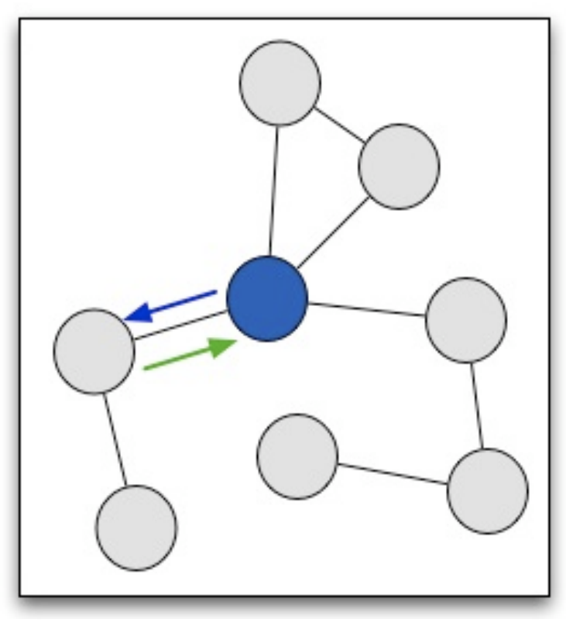
INTRODUCTION

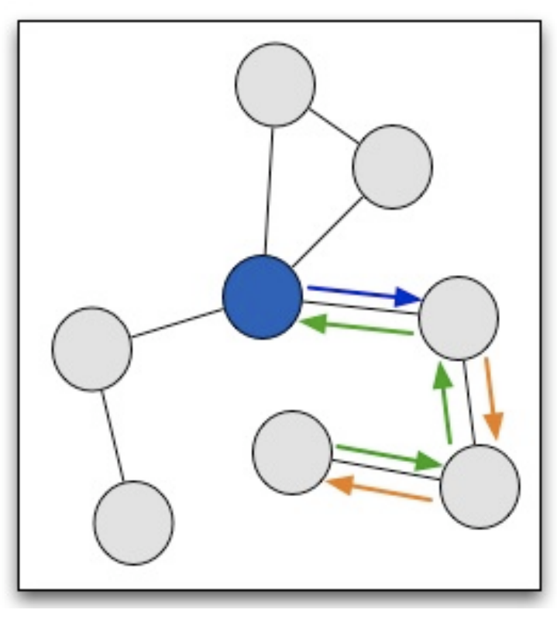
In this project, a simulation of a social system has been implemented, in which the word “tag” represents an observable social characteristic shown by the agents. Individuals, normally, tend to interact with those that present similar tags or features. With this idea, a tag-based network is created, in such a way that these similarities are expressed as links between nodes (Fig. 1). In this way, two nodes will be neighbors if they have a similar characteristics and therefore, they share a link. This relationships will determine the interactions among the nodes, as one of them will only interact with those that are his neighbors. On the other hand, this list of neighbors is not dynamic, each node will be able to change it (mutating his tag value and his behaviour), creating a different interaction group.

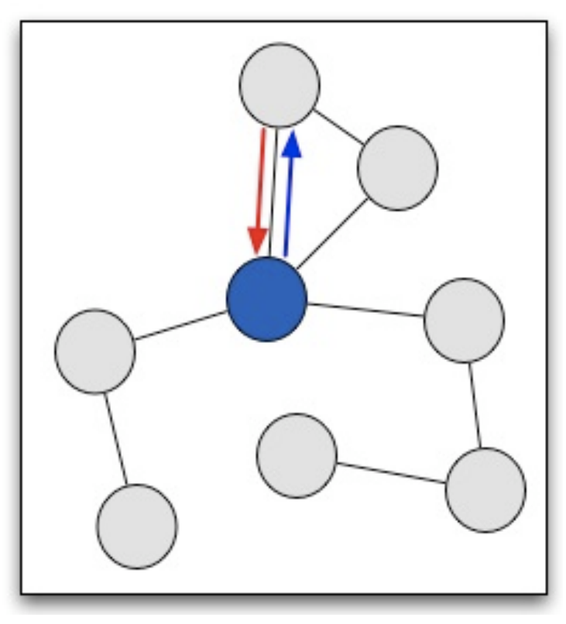
Nodes will request and answer different number of queries to and from other nodes in the network. When node ’i’ sends a request to node ’j’, then 'j' will process the query if it has some answering capacity remaining. If the query is processed, with some probability the query will be answered (Fig 2). If it is not answered, it will be passed to a random neighbor (Fig. 3); a TTL (Time To Live) value is used so the petition is not infinitely passed. In case his answering capacity is zero, it will reject the query (Fig.4). Each node gets an utility value according to the answers he has received from his neighbors.

Cheating nodes: Some nodes act in a selfish way, using all their capacity to generate new queries, not answering to other nodes’ requests.


1.- Tag-based network


2.- Accepting the query


3.- Passing the query

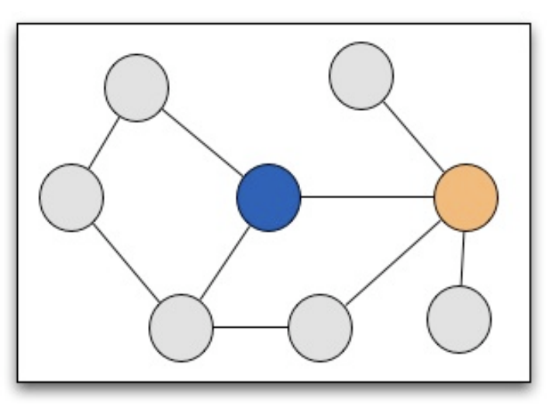

4.- Rejecting the query

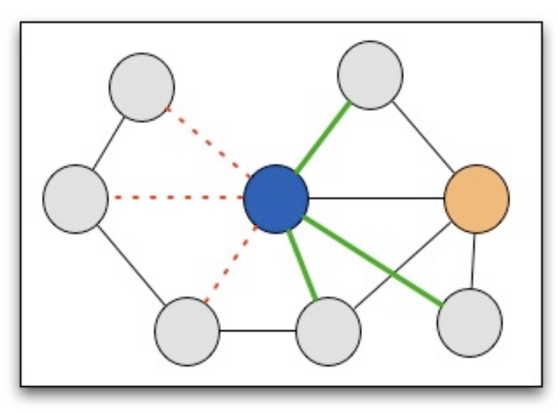
EXPERIMENTATION

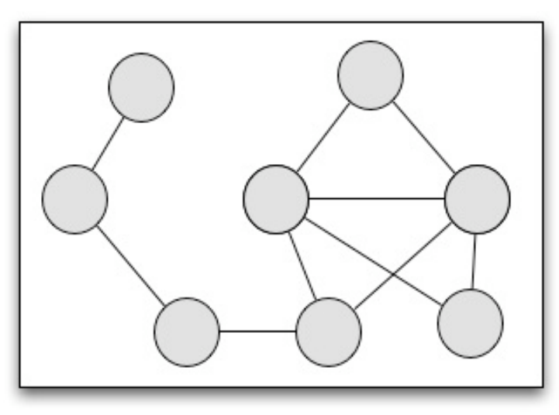
In an initial experimentation phase, the utility value for a particular node is calculated as the average of the utilities obtained from the queries that have been made by him.

Dynamic network: With a probability of ‘p’, each node compares himself with a random node from his list of neighbors (Fig. 5) and if this one performs better than him (the utility value is bigger), he will drop all his links and will copy the list of neighbors of this node.

In order to get some diversification, a **mutation** is performed with some probability. Two types of mutation have been tested: the first one deletes all links of a node and will create a random one and the second one modifies only a ’n’ number of links of that node.


5.- The node compares himself with a neighbor


6.- The node drops all his links and copies the links from his neighbor.


7.- Result after modification

History-based reputation: Each node will keep a list with the queries it has made and the results it has obtained from them. Taking into account their limited memory and giving more importance to recent petitions, only the last ’l’ results are saved. With some probability, the nodes, when receiving the queries, check this history list in order to get the utilities he has obtained from the asking node in previous interactions. If this value is bigger than a tolerance rate, the query is answered. In case no value is obtained from the history list he will ask to its neighbors to get some references. As cheating nodes are punished with this mechanism, the utility value now is calculated as the sum of the utilities obtained from all the queries the node has made, being in this way a more realistic situation.

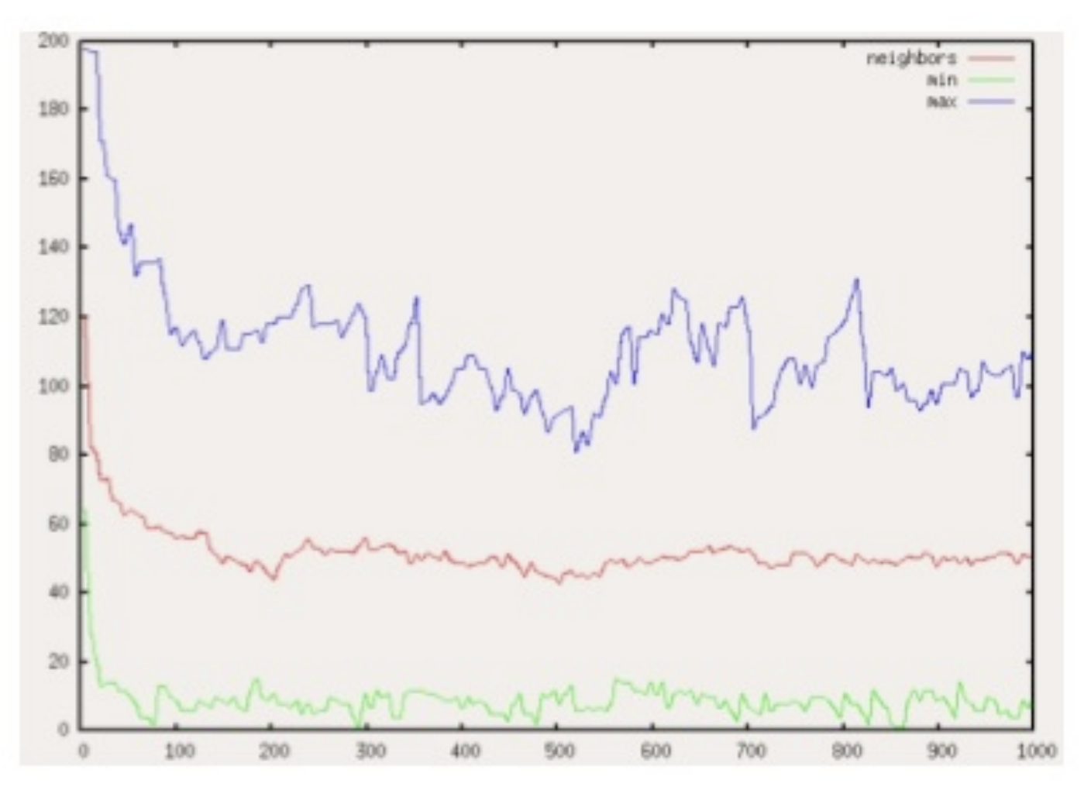
RESULTS & CONCLUSIONS

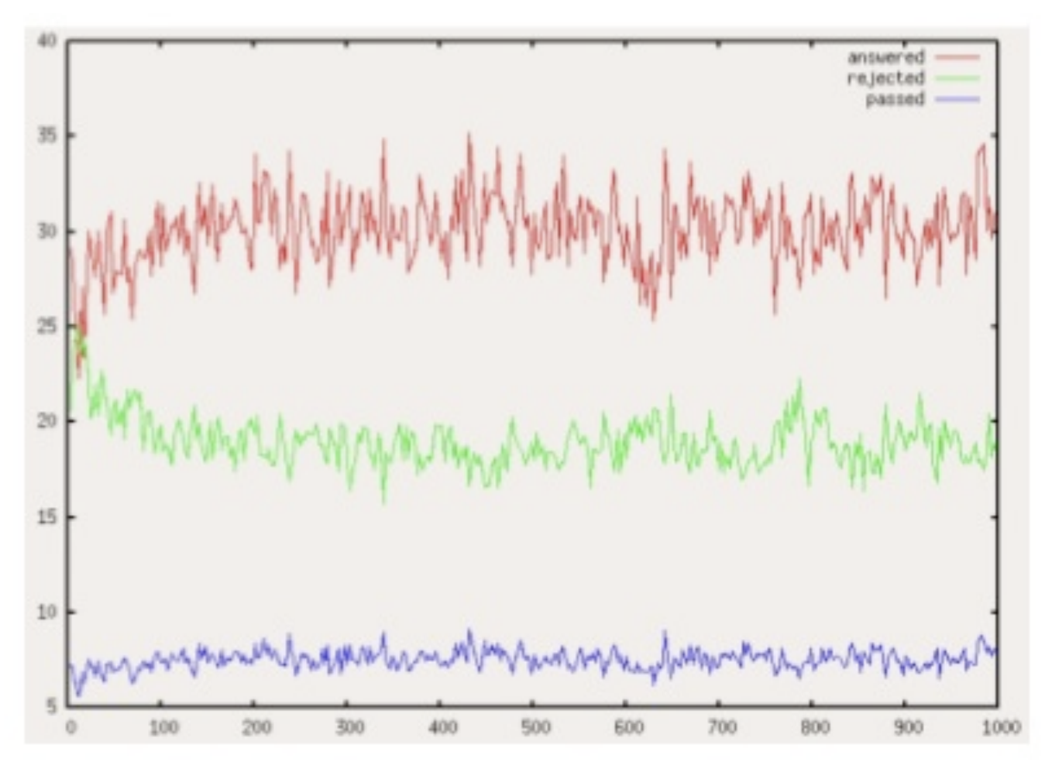
The nodes tend to decrease the number of neighbors. In a network with 200 nodes, the initial range is between 1 and 200, but after many cycles, it is reduced to between 1 and 100 (Fig.8). When an extreme mutation (all the links are dropped and a random one is created) is applied, the results neighbor list shrinks significantly and the utilities values obtained by the nodes are also worse. So, finally a less dramatic mutation, in which only a random percentage of links are changed, has a been used.

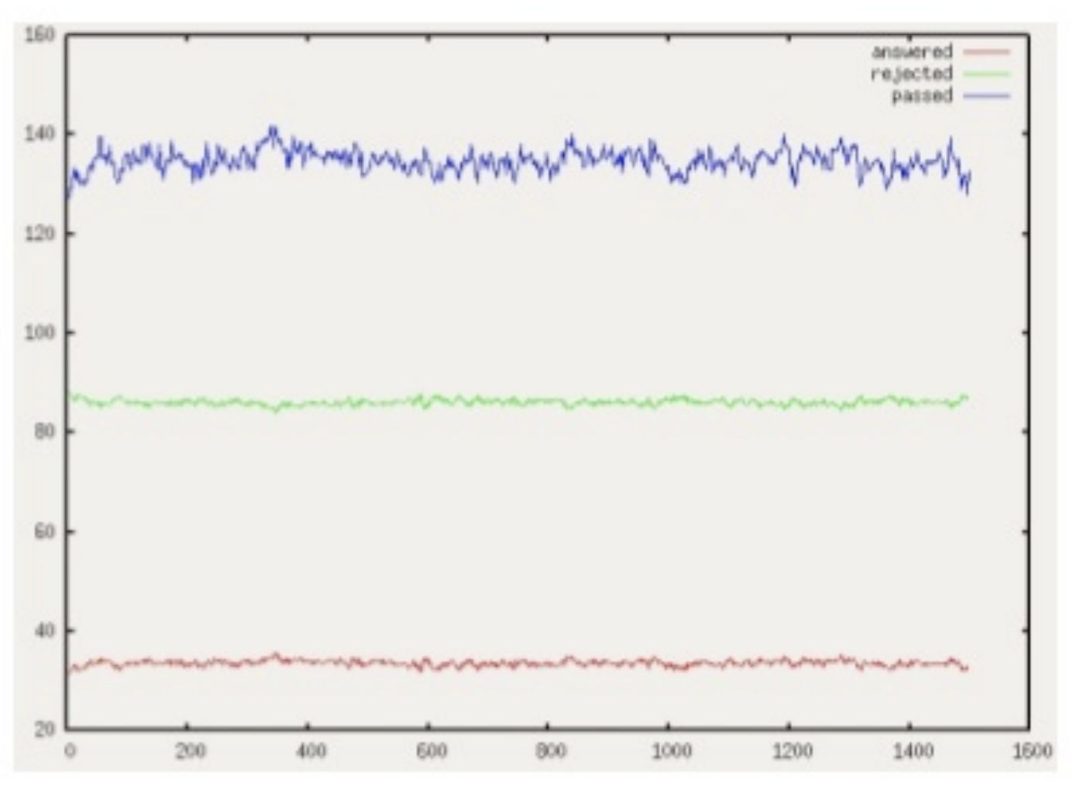
By introducing some cheaters into the network, it can be seen that the number of passed and rejected queries increases and the number of accepted decreases, as it was expected (Figs 9 & 10: answered, passed, rejected)

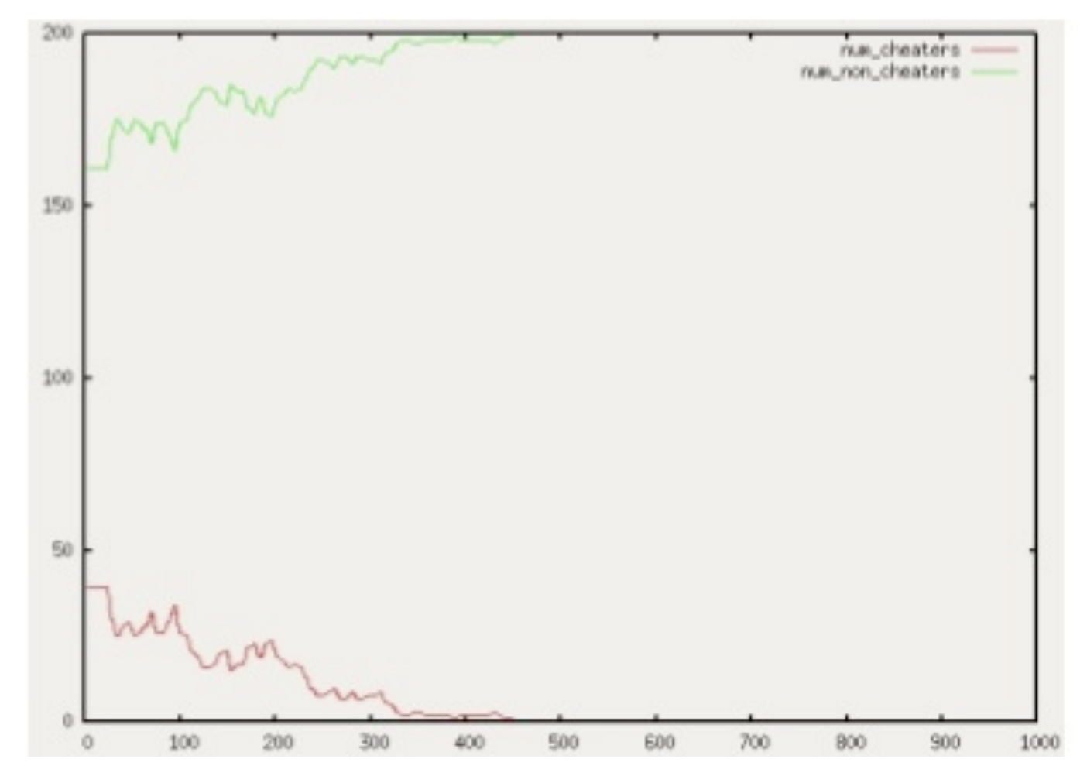
If the utility is calculated as the average of the utilities among the queries the node has made, non cheater nodes get bigger values. For this reason, if the nodes try to copy other nodes’ behaviour in order to increase their utility , the number of cheaters is also likely to get smaller, as they will becoming more cooperative (Fig. 11).

On the other hand, in a more realistic scenario, where the utility is the sum of the utilities of each query, a better performance is obtained by the cheater nodes. This situation has been improved by using a history-based reputation mechanism, as the utility of the cheaters tends to decrease, while the utility of the non cheaters tends to increase.


8.- Number of neighbors: max, average, min


9.- Result of the queries (without cheaters)


10.- Result of the queries (with cheaters)


11.- Number of cheaters and non_cheaters

URSS EXPERIENCE

In short it can be defined as an enriching and valuable experience. It has provided me with an improvement of my research skills, as an initial reading about previous and exisiting related work was needed for the project. Furthermore, as English is my second language, it also has helped me to learn and improve my writing skills. I have also gained a better capacity of implementation and experimentation, being able to analyze the results obtained. Finally, it has been a good opportunity to work in an area I´m very interested in and learn more about it

Student: **Marta Moretón Arancón**
marta.moreton@gmail.com

Tutor: **Nathan Griffiths**
Nathan.Griffiths@warwick.ac.uk

Department: **Computer science**

THE UNIVERSITY OF WARWICK