Mitigating churn in P2P storage systems

Leandro Ribeiro
Instituto Superior Técnico, Universidade de Lisboa
Lisboa

ABSTRACT

The exit and entrance of P2P network's nodes, the churn, is a widespread problem in this type of networks. P2P storage systems suffer from this problem because due to the churn there is data loss and data unavailability, which has led P2P storage systems being deprecated compared to cloud storage systems. In order to mitigate the problem of data loss or data unavailability, we will present a solution that consists of using an incentive mechanism to reward the nodes based in their reputation and a reputation system to assess the behavior of network nodes. With this reputation system we intend to motivate the nodes to be connected to the network to improve thier own reputation, give better rewards to those with better reputation, since they will be contributing to better network operation. These rewards will go through more storage on the network. Given that the network intends to offer more storage for its nodes, then it is necessary to efficiently use the storage available on the network, so we will use data compression methods to minimize the storage occupied by the data on the network.

INTRODUCTION

Peer-to-peer networks (P2P) have become popular in recent years due to the fact that they don't need a centralized server to work, such as BitTorrent which is a very popular and widely used for sharing data and digital files P2P network. In these networks, each user is a node in the network that can leave and enter relatively frequently, which means that P2P networks may have some reliability problems, as it is not certain that a given node in the network is connected at any given time. The fact that these networks do not need a central server means that there is a great availability on these networks. The decentralized organization of P2P networks makes it possible to distribute the computational load, network traffic and storage across all the nodes that be part of the network. An example of a P2P network is the case of distributed registries based on blockchain which is a network that aims to maintain secure and non-anonymous transactions over a decentralized P2P network.

According to Gantz el al.[5] there has been an exponential increase in data over the past few years, which means that there is a greater need to store data produced remotely, due to a greater number of mobile devices and their mobility. Data must be accessible from anywhere around the globe so that data is always available when needed. Data storage services (DSS) are systems that allow us to store data remotely. The DSS come to solve the problems mentioned above regarding the increase in the number of mobile devices and their mobility, in order to access data from any device anywhere in the world. Cloud DSS(cloud storage) are georeplicated DSS (spread across the globe) where physical servers with stored and replicated data are present to allow us to access them whenever we need with a high degree of assurance that our data is available when we need them. As a rule, the provider of this type of service charges us a fee for the service. DSS on P2P networks are systems that allow us to access data on any device anywhere in the world, such as cloud storage, but instead of using physical servers from a cloud service provider (cloud provider), each node of a given P2P network provides a portion of the storage of your device in order to be able to store data from other nodes. In order to make this possible, it is necessary that the nodes are connected to the network whenever its possible, to allow that when the data is accessed, it is available.

Due to the feature of P2P networks, there is no need for a central server so there is no guarantee that there will be a permanently contactable device. In P2P storage networks, each node provides a fraction of its storage to the P2P network where it will store data from other network users, and in exchange expects that its data stored on the network will be available when it needs them. In P2P networks, the entry and exit of network nodes happen relatively frequently, which leads to instability in the network and consequently can lead to the problem of a given node having data from others and suddenly leaving the network. This can lead to a user who had the data saved on the network losing his data, as another who had his data saved ceased to belong to the network.

There is a lot of diversity of nodes connected to P2P networks. Some of these nodes have a better connection to the Internet or faster Internet than others, while some may spend more time connected to the network. The quality of network access, the time that a node spends connected to the network and the number of inputs and outputs of the network node are features linked to the churn, while the speed, quality and storage space that a node offers to the network, are features linked to storage.

On the one hand, given that there is churn in storage systems, we need replication for availability. On the other hand, the effects associated with churn disturb the total available space, and it would be important to know which network nodes leave and enter this network with some frequency, in order to decrease their guarantees and storage space. The process of identifying nodes with a high degree of churn can be done using a reputation system that allows nodes to know if other nodes are reliable regarding their stability in the network, the churn. The reputation system will allow us to evaluate the behavior of other nodes in the network so that the network knows the behavior of a given node. To assist the reputation system there will be an incentive system that will offer better rewards depending on the better reputation of a given node, so that the nodes will be motivated to have a better reputation in order to obtain better rewards and thus the nodes end up for reducing your churn so that the network can trust more on these nodes, while nodes with worse reputation will have less storage space on the network.

In this thesis a protocol was developed for DSS's P2P that mitigates churn and its impact on the data stored in the system, both in terms of unavailability and in terms of the data being kept in the system and not being lost. In the evaluation of the protocol, a P2P DSS was implemented with this protocol, then simulations were developed that explored mechanisms of churn and other configurable data that can simulate the behavior of users. Using the incentive and reputation mechanism, incentives are offered to nodes and they are evaluated and data compression techniques are used in order to reduce their size to obtain better efficiency in storing them and an adequate quality of service on the network.

Document Structure

The document is organized as follows. Section 2 presents the related work, in Section 3 the incentive and reputation mechanism, in Section 4 the evaluation methodology and the evaluation of the proposed system and Section 5 presents the conclusions and future work.

RELATED WORK

A P2P network is a distributed network where all its participants share part of their hardware resources. These shared resources are accessible by all nodes in the network without going through intermediate entities, Schollmeier et al.[8]. P2P networks are different from Client/Server networks mainly because there is no central entity, since in P2P networks there is the concept of Servent, in other words, each node in the network acts as a server and client at the same time.

A P2P storage system has the objective of making the nodes of a given P2P network provide their own storage for the network. The network in its turn shares this storage available to all its participants.

According to Dabek et al.[4], a P2P storage system aims to make the nodes of a given P2P network available storage for the network. In order for all participants to have storage for their data, the network in its turn shares that storage available to all its participants.

Chord [9] is a distributed search protocol that solves the problem of efficiently locating a network node that has a particular object. This protocol supports a single operation: given a key, it maps the key to a node. Depending on the application that uses this protocol, that node may be responsible for storing a value associated with that key. Chord simplifies the design of applications and P2P systems based on its protocol, addressing the following difficulties:

- Load balancing
- Decentralization
- Scalability
- Availability
- Flexible Naming

Eyo - Device-Transparent Protocol [10] is a P2P personal storage system that aims to provide transparency to the user in the face of disconnected devices. Eyo synchronizes updates between all devices. Eyo divide metadata from the content of the object and replicates the metadata across all devices so that each device has the ability to manage each object even without having its content. In order to search for a particular object, Eyo supports a query system (queries) in order to search for objects using metadata. Eyo supports automatic conflict resolution, for that there is a version history mechanism allowing for when the devices synchronize a given object and different versions appear, it's possible to check which version overlaps the others.

WheelFS [11] is a distributed storage system to help applications share data and achieve fault tolerance. WheelFS is resistant to Byzantine failures. This system provides primary/backup file replication, keeping a copy on others servers in case the primary fails or is unavailable at the desired time. WheelFS also allows customers to cache so that other customers can directly access other customers' cache to retrieve some file.

PRACTI [3] is a solution for large scale replication, it provides 3 properties to the applications that use it. These properties are the following:

- Partial replication that consists of the system allowing to replicate part of the data or metadata on any node in the system.
- Arbitrary consistency that allows applications to choose whether to use strong or weak consistency.
- Independent topology which allows any node to exchange updates with any node as in other existing P2P systems.

PRACTI has a log which is where all modifications made to the local node are kept and then updated in Checkpoint in order to support the partial replication policy. It is in Checkpoint that changes are stored in order to ensure consistent causal view of system data. Thus allowing each node to store any data set locally and change at any time.

SYSTEM

The objective of this dissertation is to develop a DSS P2P that mitigates churn and its impact on the data stored in the system, in terms of unavailability and in terms of data persistence in the system. For this, we will use mechanisms of incentive and reputation to evaluate the nodes of the system and to combine with these mechanisms, data compression techniques in order to reduce the size of the data to obtain better efficiency in the storage of these. In this system, each user will be able to store data on the network and, in return, each participant will offer storage to the network so that there is storage available so its possible for others to store their data. In the system, participants can perform some operations such as storing objects (writes), obtaining their objects (reads) and removing their objects (removes).

The reputation and incentives mechanism presented have the goal to motivate users to spend more time connected to the network and so mitigate churn and, consequently, its effects, such as data unavailability and definitive loss of these. In order to motivate the participants to not leave the system, their permanence in the system will be measured, as well as the operations carried out successfully and offering more storage space to the nodes that better cooperate in the storage service. To use these mechanisms in a given system, it is necessary that this system is implemented in a P2P network and that it is a data storage system. This system must allow perform writes, reads and removes in the network. To improve the efficiency of the network and the proposed mechanisms, it is advisable to use data compression in order to reduce the size of the objects for a better efficiency in their storage.

The fault model considered is the crash model. In this model, the nodes can fail due to stop, it's no longer possible to contact them through the network or the nodes themselves can no longer contact other nodes through the network. These failures can be permanent if a node fails and never recovers from the failure or may be temporary, that is, the node eventually recovers from the failure and returns to the network. We assume that all participants in the system don't suffer from Byzantine faults.

Incentive and reputation mechanism

In order to reduce the churn we want to encourage participants to spend more time connected to the network. For this we will offer incentives for them to spend more time connected to the network. In this subsection we will present the proposed mechanism of reputation and incentives.

The storage system is based on contributions from the participants as the total network space is the space offered by all nodes in the system, it's necessary to use the storage space efficiently. For this reason, the reputation system is complemented with data compression techniques in order to make storage efficient.

One node, when it joins the network, commits to store and supply a particular object when requested by its owner and, in return, also expects it to be able to store objects on the network and get them back when it needs them. The operations performed by the network nodes can be completed successfully or they can fail. In the case of a writes, it can be successfully completed if the object is stored or it can fail if the object is not stored. In the case of reads, it is successfully completed if the desired object is returned, or it fails if no object is returned. The results of these operations can be measured according to their success and thus evaluate the behavior of their participants.

In order to evaluate the reputation of a node, a reputation algorithm is used, based on the Eigentrust[6]. In the Eigentrust algorithm, positive and negative transactions between all nodes in the system are counted, while in the proposed algorithm, the time spent by a node connected to the network and the time its absent from the network are counted. This adaptation is made because it is intended to evaluate the availability of nodes in the system instead of the transactions made between them. This is accounted by a ping mechanism, where each node periodically communicates with one node or a group of nodes that are responsible to calculate the reputation (could be the entire network) in order to show that it is connected and present on the network. After a set of pings fails, the node is assumed to have failed. These nodes that is responsible to calculate reputation store the pings of the node *i* successfully made suc(i, j) and the pings that weren't made are called unsuccessful insuc(i, j). As in the algorithm identified above, the value s_{ij} defines the differences between successes and failures and is defined by:

$$s_{ij} = suc(i,j) - insuc(i,j) \tag{1}$$

In order to aggregate the local confidence values, as suggested in the algorithm, it is necessary to normalize these values. The normalized values are called normalized confidence values and are given by the following formula:

$$c_{ij} = \frac{max(s_{ij}, 0)}{\sum_j max(s_{ij}, 0)} \tag{2}$$

The sum of all c_{ij} of a *j* is 1. Periodically each node does reputation calculations based on the c_{ij} values. The reputation value is always a positive value. To calculate reputation, it is necessary to calculate the mean ($\bar{x}$) and the standard deviation (σ) of the values of c_{ij}. The reputation value of the node *i*, r_i, is given by the following formula:

$$r_i = \frac{max(c_{ij} - \overline{x}, 0)}{\sigma} * 100 \tag{3}$$

The reputation values are stored by the respective node that performed the calculation. Reputation values are values greater or equal than zero. Incentives are offered to those with a better reputation. These incentives include the amount of storage offered to each node. This amount of storage offered to each node is relative to the storage it offered to the network. The Table 1 returns the amount of relative storage given the node reputation. There are no negative reputation values due to the Formula 3. In the Table 1 the left column

represents the possible reputation values that the nodes can take while the right column represents the rewards they will obtain according to their reputation.

The pings made by the nodes are sent to any node that is responsible to calculate the reputation. These, in their turn, share with each other node that are responsabile to calculate the reputation the pings received in order to ensure that the values are consistent with each node. After aggregating the received values of pings, each calculates the reputation value of each node. Since the values will all be synchronized, the participant's reputation values will be the same among all nodes that are responsible to calculate reputation.

Reputation	Rewards
0	50%
0-25	75%
25-50	100%
50-100	125%
100 - 150	150%
150 - 200	175%
200+	200%

Table 1. Reputation/Rewards table

According to Piatek et al.[7], incentives discourage free-riding behavior. So with the use of incentives, users tend to improve their behavior and thus lead to a reduction in their churn and the effects it caused.

For more efficient storage of data in DSS P2P, it's recommended to use data compression in order to achieve a better use of the available space and thus a better quality of the service provided.

EVALUATION

In this chapter we will present the evaluation of the proposed mechanisms to reduce churn. The purpose of this evaluation is to show the proposed mechanisms are able to motivate users to reduce churn and consequently mitigate its impact. For the evaluation of the system we will perform some simulations using data that simulate the behavior of users of P2P networks to produce access patterns to the storage systems. In Section 4.1 we describe the data we use in the simulations and, in Section 4.2, we present a single simulation performed without the support of the reputation protocol that serves for present the system and show some of the results obtained. In Section 4.3 we evaluate the system using the reputation and incentives protocol against a system without these mechanisms. In Section 4.4 we evaluate the system using the incentive and reputation protocol, this time without using data compression and comparing the system without using this mechanism. In Section 4.5 we conclude this chapter with the results from the evaluation performed.

Simulations description

To perform the simulations, we use Corten simulator in order to obtain results that allow conclusions to be drawn about the performance of the proposed mechanisms. To carry out

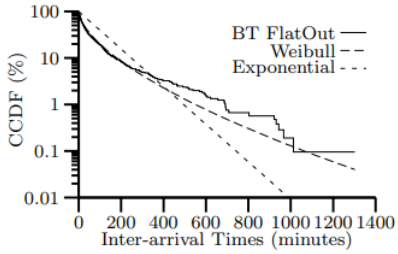

Figure 1. Time between the entry of nodes in P2P networks

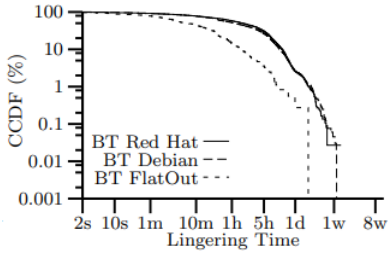

Figure 2. Node's session lenght in P2P networks

the simulations, the system described in Chapter 3 was designed using the Rust language. The simulations use parameters taken from real studies that observed the behavior of users of file systems.

To fit the network entries with real data, each node enters the network after another node with a time interval taken from a Weibull distribution where the shape parameter is k = 0.62. Fig. 1 shows data taken from a study by Stutzbach et al.[12] where the time between the entry of nodes in the network corresponds to the referred distribution.

The session sizes of the nodes connected to the P2P network are also obtained from the same study and are taken from a Weibull distribution with k = 0.59, lam = 41.9, shown graphically in Fig. 2, as shown in [12].

Fig. 3, taken from the study by Stutzbach et al.[12], shows the time that the nodes are out the network. In the simulations performed, values from this table are used so that the simulation has this approximation to the study referred.

In the simulations that were carried out, each node has an amount of storage available for storage on the network. When a node enters the network, it defines the amount of storage.

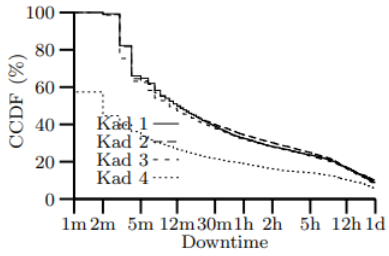

Figure 3. Time that's a node is absent from P2P networks

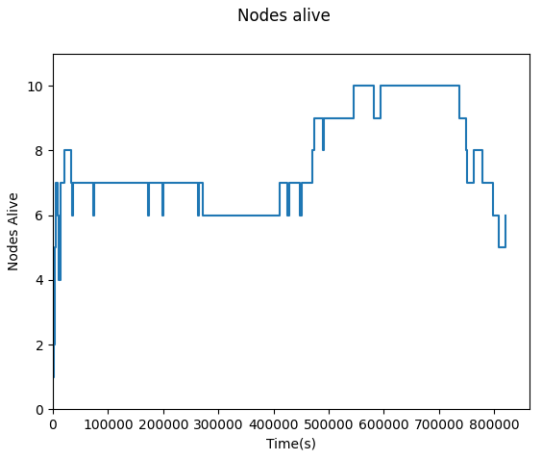

Figure 4. Number of active nodes in the simulation

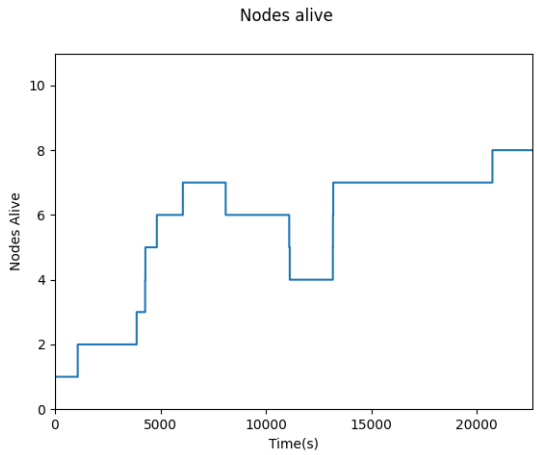

Figure 5. Node's entry at the start of the simulation

This value is taken from a log-normal distribution where the value μ is 16.9 and the value σ is 1.0 which leads to an average storage of 36GB offered by each node in the system as suggested by the study by Anderson et al.[2].

A study realized over 5 years on file systems by Agrawal et al.[1] shows that the file size increases on average 15% per year. In 2004, the year of the study, the average file size was 189KB. At the date of the simulations, taking into account the evolution proposed by Agrawal et al.[1], the average file size will be approximately 1769KB. In the simulations performed, the objects created and stored by the system's nodes are removed from a log-normal distribution where the value μ is 6.98 and the value σ is 1.0. This distribution average that the files created have an average of 1769KB.

System simulation

In this section we are going to present a simulation in order to show the system and the results.

In the simulation presented in this section, 10 nodes are used in the network. During the course of this simulation, 9550 objects were written to the network. Fig. 4 shows the number of nodes that are alive at any given moment. We can observe the general behavior of the network nodes during the course of the simulation, and we can observe some instability in the network with some outputs and inputs of the nodes which can lead to loss or unavailability of data, which ends up not happening in this simulation because it is a short simulation with few nodes. In Fig. 5 we can observe the initial phase of the network while the nodes are entering for the first time and that while some nodes are still entering the network, others have already left and re-entered.

Simulations between systems and results comparison

In this section we will present the results of the simulations of a P2P storage system based on the Chord protocol as well as the same system with the addition of the proposed incentives and reputation mechanisms. Both systems use the same simulation, that is, the nodes in the system have exactly the same behavior, each node leaves and enters the network at exactly the same time in both simulations. Ahead, the system without a reputation and incentives mechanism will be called System1 while the system with a reputation and incentives mechanism will be called System2. The simulation with System1 will be called Simulation1 while the simulation with System2 will be called Simulation2. Due to a problem presented by the simulator used that failed when the number of events is very high, it is not possible to perform simulations with more events than the simulations presented here, as it presents a failure if the simulations are very extensive and if it simulate a high number of nodes.

In the simulations in this section, the time between the node entrances in the network was reduced, the Weibull distribution was used as explained above but with a smaller entrance interval to allow simulations to be performed with a larger number of nodes in the time that is possible before the simulator fails. The reduction of this interval between the entry of nodes in the network can lead to the existence of some errors in the reading of objects due to the rate of entry of the nodes in the network, which may lead to the Chord taking some time to stabilize the ring and consequently objects are still being transferred between nodes. This problem can occur mainly in the initial phase of the simulations, which is when there is a greater number of nodes compared to the nodes in the network. This problem can be solved by redoing the request to read the same object after a short time.

In both simulations performed, 6333 nodes entered the network. Fig. 6 shows the number of nodes that are in the network during the simulation. In this graph we can see that the nodes enter the network as the simulations advance in time and they can leave and return to the network, this behavior is in accordance with that described in Section 4.1, at the final moment of the simulation they are approximately 70 % active nodes in the network.

During both simulations, all nodes perform an operation every 10 seconds. This operation can be either a write or a read of any object, where a write have the same probability to ocurr than a read. The idea of these simulations is to saturate

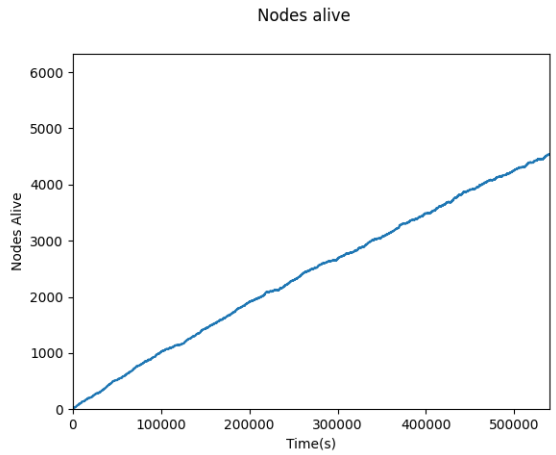

Figure 6. Number of active nodes in Simulation1 and Simulation2

Figure 7. Writes errors occurred during Simulation1 and Simulation2

the network with objects and we can quickly fill the network, therefrom no object removes are made by the nodes.

During Simulation1, nodes attempted to write 6 466 046 objects, where approximately 79.96% of these were successfully performed. During Simulation2, nodes attempted to write 6 296 083 objects, where approximately 92.20% of these were successfully performed. Fig. 7 shows the graph with the evolution of the number of write errors that occurred during Simulation1 and Simulation2. In this graph we can see that there is a smaller number of write errors in Simulation2 compared to Simulation1, this is mainly due to the smaller space occupied by the discs in Simulation2 that is obtained due to the compression of the objects used.

During Simulation1, nodes attempt to read 5,743,644 objects on the network, where approximately 99.95% were successful. During Simulation2, nodes attempt to read 5 827 433 objects on the network, where approximately 99.96% were successful. Fig. 8 shows the graph with the number of read errors that occurred during Simalation 1 and Simulation2. Some of these errors can happen due to the change in node objects that occurs after the entry or exit of nodes in the network. This

Figure 8. Reads errors occurred during Simulation1 and Simulation2

problem can be mitigated if the same read request is made after a period of time. However, there may also be cases where objects are permanently lost or are effectively unavailable, if the 3 replicas that keep copies of a given object are absent from the network and it is not possible to move those copies to other nodes or create additional copies. The difference between the number of errors is almost none. This is due to the fact that replication factor 3 was used, that is, in addition to the original copy of the object, two additional replicas are stored. Due to the use of this replication factor, there is more storage used by the network to store the object, but on the other hand, the probability of an object being permanently lost or that there is a temporary loss is relatively less.

Simulations between systems without compression and comparison of results

In this section we will present the results from the simulations of a P2P storage system based on the Chord protocol and the results of the simulations of the same system with the addition of incentive and reputation mechanisms, this time without the use of data compression. As in the previous section, both systems use the same simulation or, in other words, the nodes in the system have exactly the same behavior, leaving and entering the network at the same time of the simulation. In this section, the system using the reputation incentive mechanism is called System3, while the system without using this mechanism is called System4. In turn, the simulation with System3 is called Simulation3 while the simulation with System4 is called Simulation4.

In the simulations presented in this section, 3000 nodes were used. During the execution of the simulations in this section, the time between the entrances of the nodes in the network was reduced, the Weibull distribution was used as explained above, but with smaller input intervals, this time 50 times smaller, in order to distinguish the input phase nodes to the phase where the nodes are already in the network. Due to the shortened entry time, we can observe a large initial entry rate up to the peak of the number of nodes. After this moment during the simulation, the number of active nodes in the network will tend to a number around 2000 active nodes, which would

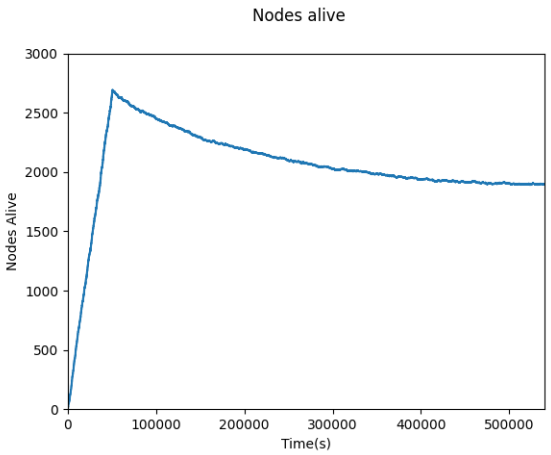

Figure 9. Number of active nodes in Simulation3 and Simulation4

Figure 10. Reads errors occurred during in Simulation3 and Simulation4

be the same number that would be tended if this shortening was not used.

As in the simulations in the previous section, during both simulations all nodes perform an operation every 10 seconds. This operation can be either a write or a read of any object, with both having the same probability of occurring. The idea of these simulations is to saturate the network with objects and thus quickly fill the network, therefrom no object removes are made by the nodes.

During Simulation3, nodes attempted to read 4,711,189 objects in the network, where approximately 99.91% were successful. In Simulation4 there were 4,091,759 attempts to read objects where 99.87% were successful. In Fig. 10 we have the number of read errors per hour that occurred in Simulation3 and, in Simulation4, it is represented graphically. In this simulation, a replication factor 2 was used, that is, in addition to the original stored object, an extra copy of the object was stored. We can see that there was a higher rate of reading errors in the simulations in this section (Simulation3 and Simulation4) compared to the simulations in the previous section (Simulation1 and Simulation2). But despite this slight increase in the number of errors, less space is consumed to store objects and their replicas, which frees up extra storage on the network so that it can be used by network nodes to store more objects.

During Simulation3, nodes attempted to write 5 120 896 objects in the network, where approximately 69.69% were successful. In Simulation4, nodes attempt to write 5 438 844 objects in the network, where 63.34% were successful. In Fig 11 are represented graphically the writing errors that occurred during the execution of simulation3 and Simulation4. We can observe a lower error rate and a smaller number of errors in Simulation3. This is due to the fact the nodes with less reputation have less storage available which leads to a smaller total number of written objects.

In Fig. 12 we have represented graphically each of the levels of the reputation and incentives in the Table 1 and the number of nodes that are at each level throughout the Simulation3.

Figure 11. Write errors occurred during in Simulation3 and Simulation4

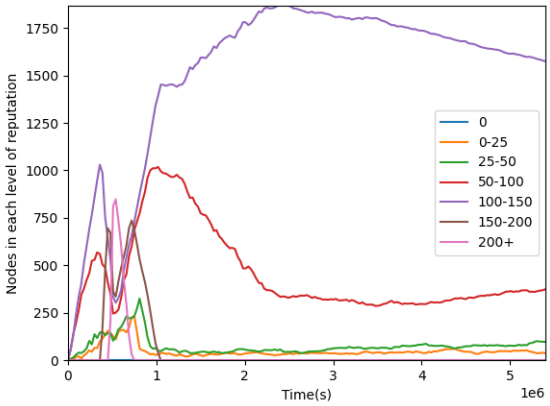

Figure 12. Graph with the distribution of nodes by the levels of reputation/reward

Results discussion

Piatek et al.[7] concludes that incentives discourage the behavior of free-riding by the nodes of the P2P networks. With the results obtained in this chapter we can prove that the use of the proposed mechanism helps to reduce the effects caused by the churn. Which leads us to believe that the use of this mechanism leads the nodes to reduce their churn. Since in all simulations performed with the same data from real studies that observed the behavior of users of file systems were used, it wasn't possible to evaluate the behavior of users. But a real implementation of these mechanisms would benefit network users.

The use of the data compression mechanism is beneficial for the system since with data compression it's possible to reduce the size of the data on the network and so offer more incentives to users which could further motivate them to reduce the churn.

CONCLUSION

P2P networks are seriously affected by the churn and DSS P2P networks are no exception. In these systems, due to the churn, the permanent or temporary exit of nodes leads to data loss or temporary data unavailability.

In P2P networks there is a behavior called free-riding, which is when users consume system resources without contributing to it. We saw that we can mitigate this behavior by offering incentives to those who have better behavior. For this, users try to behave better and then reduce their churn and mitigate free-riding.

In this work, a protocol was designed to mitigate churn and its effects. This protocol aims to offer incentives to mitigate the churn. For this, the nodes belonging to the network are evaluated according to their connection to the network or, in other words, how long they spend connected to the network. In this way it's possible to know which nodes have the best behavior. Next, better rewards are offered depending on node's reputation. For a better operation of the proposed protocol it is recommended to use a data compression mechanism system in order to take advantage of the available storage on the network more efficiently.

The protocol was evaluated through simulations. To perform the simulations a P2P simulator was used. In order to evaluate the protocol, it was implemented in an DSS that followed the structure of the Chord[9], so, the entire internal organization of the nodes, dissemination of messages, entry of nodes in the network, distribution of data by network is in accordance with the Chord.

Future work

As a future work, the protocol can be improved in order to assess, depending on the behavior of users, to be able to vary the reputation/benefit table in order to adjust the rewards with the general behavior of the network, so that it will be possible to offer a good quality of service to users and offer even better rewards to further motivate users to improve their behavior. Another interesting feature that can be developed for the protocol is to allow that through the reputation of the nodes, the protocol can estimate how many replicas of an object must be created according to the reputation of the nodes that store copies of the object, so that object isn't lost permanently or is unavailable for a period of time due to the nodes that store replicas of a given object have left the network (permanently or temporarily). In this way it is possible to free up even more storage for the network and then increase the rewards offered to users, while continuing to offer a good quality of service.

In addition, a future version of the protocol should allow the protocol to support Byzantine failures of the nodes belonging to the network, no longer supporting, exclusively, crash type failures.

REFERENCES

1. Agrawal, N., Bolosky, W. J., Douceur, J. R., and Lorch, J. R. A five-year study of file-system metadata. *ACM Transactions on Storage (TOS) 3*, 3 (2007).
2. Anderson, D. P., and Fedak, G. The computational and storage potential of volunteer computing. In *Sixth IEEE International Symposium on Cluster Computing and the Grid (CCGRID'06)*, vol. 1, IEEE (2006), 73–80.
3. Belaramani, N. M., Dahlin, M., Gao, L., Nayate, A., Venkataramani, A., Yalagandula, P., and Zheng, J. PRACTI Replication. In *NSDI*, vol. 6 (2006), 5–5.
4. Dabek, F., Kaashoek, M. F., Karger, D., Morris, R., and Stoica, I. Wide-area cooperative storage with CFS. In *ACM SIGOPS Operating Systems Review*, vol. 35, ACM (2001), 202–215.
5. Gantz, J., and Reinsel, D. The digital universe in 2020: Big data, bigger digital shadows, and biggest growth in the far east. *IDC iView: IDC Analyze the future 2007*, 2012 (2012), 1–16.
6. Kamvar, S. D., Schlosser, M. T., and Garcia-Molina, H. The Eigentrust algorithm for reputation management in P2P networks. In *Proceedings of the 12th international conference on World Wide Web*, ACM (2003), 640–651.
7. Piatek, M., Isdal, T., Anderson, T., Krishnamurthy, A., and Venkataramani, A. Do incentives build robustness in BitTorrent. In *Proc. of NSDI*, vol. 7 (2007).
8. Schollmeier, R. A definition of Peer-to-Peer networking for the classification of Peer-to-Peer architectures and applications. In *Proceedings First International Conference on Peer-to-Peer Computing*, IEEE (2001), 101–102.
9. Stoica, I., Morris, R., Karger, D., Kaashoek, M. F., and Balakrishnan, H. Chord: A scalable Peer-to-Peer lookup service for internet applications. *ACM SIGCOMM Computer Communication Review 31*, 4 (2001), 149–160.
10. Strauss, J., Paluska, J. M., Lesniewski-Laas, C., Ford, B., Morris, R. T., and Kaashoek, M. F. EYO: Device-Transparent Personal Storage. In *USENIX Annual Technical Conference* (2011).

11. Stribling, J., Sovran, Y., Zhang, I., Pretzer, X., Li, J., Kaashoek, M. F., and Morris, R. T. Flexible, Wide-Area Storage for Distributed Systems with WheelFS. In *NSDI*, vol. 9 (2009), 43–58.

12. Stutzbach, D., and Rejaie, R. Understanding churn in Peer-to-Peer networks. In *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, ACM (2006), 189–202.