A Diffusive Load Balancing Scheme for Clustered Peer-to-Peer Systems

Ying Qiao
School of Information Technology and Engineering
University of Ottawa
Ottawa, Canada
yqiao074@site.uottawa.ca

Gregor v. Bochmann
School of Information Technology and Engineering
University of Ottawa
Ottawa, Canada
bochmann@site.uottawa.ca

***Abstract*—Node clustering is an effective solution for achieving good performance and high reliability for peer-to-peer (P2P) systems. To improve the performance of a clustered P2P system, it is important to balance the service load among the clusters in the system. In this paper, we describe a diffusive load balancing scheme for clustered P2P systems, which dynamically adjusts the size of the clusters, by moving nodes among the clusters, based on their service demands and node resource capacities. Our simulations show that the proposed load balancing scheme significantly improves the performance of a P2P system in terms of balanced available capacity.**

Keywords-Load balancing; diffusive load balancing; peer-to-peer systems; distributed algorithms; dynamic resource allocation; performance management; server clusters; clustered peer-to-peer systems

I. INTRODUCTION

P2P systems become popular with their scalable architecture, flexible organization and low cost; one question is whether these systems could host services, for example, web services. Services are provided by servers in a client/server system; it is important for a server to assure the quality of its service to its clients. Because the nodes in a P2P system are highly dynamic on their on-line time and widely heterogeneous on their resource capacity, currently, P2P are mainly used for applications that are able to tolerant interruptions, e.g., file sharing, downloading, or video delivery. The ability of a P2P system to consistently provide high quality services is questionable.

Node clustering is an effective approach that can achieve high reliability and managed performance in a dynamic P2P system. However, currently, the resource capacity of clusters has not been considered in the design of clustered P2P systems, while it is possible that some service requests experience long delays while resources in other clusters are idle.

Load balancing has been proposed to fairly distribute service requests to resources and thus improve the overall performance of a system. By using an effective load balancing scheme, a system can balance the load among nodes, and the failure rates of its service requests are largely reduced [2].

Intuitively, the performance of a clustered P2P system can be improved with load balancing. The load balancing should be performed at both the intra-cluster level and the inter-cluster level, hence, not only the response times of different requests for the same service (serviced by different nodes within a given cluster) would be balanced, but also requests for different services (serviced by nodes in different clusters) would have similar response times. As the intra-cluster load balancing has been intensively studied, this paper focuses on inter-cluster load balancing and proposes a diffusive load balancing scheme, called directory-initiated scheme, for reorganizing nodes in the clusters according to their capacities and the loads experienced by the clusters.

In the proposed scheme, each cluster works as a directory and periodically checks the load statuses of its neighboring clusters. According to this collected information, a cluster could balance the loads in its own neighborhood by moving nodes from lightly loaded clusters to heavily loaded clusters; in this way, a heavily loaded cluster would obtain a higher capacity to serve its requests. As neighborhoods of clusters overlap and cover the whole network, global load balancing will be achieved through such local balancing within each neighborhood, such that all clusters will obtain a processing capacity corresponding to the load of their service requests, and the overall service quality will be uniform.

The remainder of the paper is organized as follows. Section 2 and 3 review related work on load balancing for P2P systems and discuss the benefit of load balancing in a clustered P2P. Section 4 discusses in detail the proposed load balancing procedure. Section 5 presents simulation results to evaluate the performance of the load balancing scheme. Section 6 contains the conclusions.

II. RELATED WORK ON DYNAMIC LOAD BALANCING IN P2P SYSTEMS

Load balancing faces challenges coming from the characteristics of P2P systems. First, the size of a P2P system is large. To deal with the large size, distributed architectures, e.g., tree [7], distributed directory [2], random probing [8], and skip list [9], are proposed to achieve global balance.

Second, the nodes in a P2P system are not replicas and requests cannot be executed in any node. To perform load movement, P2P load balancing techniques may place nodes among the ranges of the object space (node

placement) [8, 9] or place objects among the ranges of the node space (object placement) [2, 6, 7].

Third, a P2P system is a dynamic system with churn. It requires that the load movements should be determined dynamically according to the current load. Dynamic load balancing techniques, whose decision components collect the load status of the system and, from time to time, make load balancing decisions, can capture well the dynamics of P2P systems [2, 7, 8, 9].

Some load balancing schemes require building extra associations on top of the overlay networks. A *k-ary* tree [7] requires *(n-1)* connections for aggregating and disseminating load statuses, and a skip list [9] uses a total of $(3n - 2 - \log_2 n)$ connections for ordering nodes at multiple levels according to their load statuses. These connections are maintained during the life time of the load balancing procedure requiring extra messages and processing power. Also, when the overlay network is experiencing churn, nodes and connections of the overlay network are highly dynamic. The load status reports received at a decision component could become stale or incorrect because of these fast changes. This could directly affect the performance of load balancing.

Therefore we propose a scheme using existing connections in a clustered P2P overlay network, where each cluster balances the load in its neighborhood. This eliminates the messages for maintaining extra connections and also improves the performance of load balancing when the system is experiencing churn.

III. CLUSTERED P2P SYSTEM ARCHITECTURE FOR LOAD BALANCING

Clustered P2P systems may be organized in different ways: the nodes in the system are grouped into clusters, which in turn may be organized according to different topologies, such as tree, hypercube, or just as a flat cluster.

Different clustering architectures could be adopted for improving the reliability of a P2P system. [3] proposed a multidimensional hypercube P2P with self repairing, where a node in the hypercube is composed of multiple computer nodes and could be considered a cluster. Working synchronously, each hypercube node (cluster) balances the number of computer nodes sequentially with its neighbors. The hypercube will synchronously split/merge its nodes when the total number of its computer nodes is over/under a certain threshold. eQuus [1] connects clusters with an overlay DHT similar to Pastry. A cluster will split/merge when its size is over/under the threshold.

As we indicated before, from the point of view of resource management, it is not efficient for a clustered P2P system to arrange computer nodes into clusters according to the size of the clusters or of the system. We propose a load balancing technique to organize clusters according to the amount of service requests and the capacity of their nodes for this kind of system.

One question here is whether we can apply a load balancing scheme studied in distributed computing systems for clustered P2P systems. As P2P systems are distributed computing systems, one may argue that the load balancing techniques, including architectures and algorithms, could always be applied to P2P systems. Hence, this is the same case for load balancing in clustered P2P systems.

Another question is what insight we could gain from load balancing in this kind of system, and what adjustments should be adopted. These are the intentions of our research.

IV. DIFFUSIVE LOAD BALANCING FOR P2P SYSTEMS

We adopt a diffusive load balancing scheme here. Diffusive schemes were originally studied for massively parallel systems, e.g., distributed memory multiprocessor system, or parallel processing system; these systems have thousands of computing components.

With diffusive schemes, each computing component works as a decision component, either synchronously [10, 11] or partially asynchronously [4, 5, 12, 14], to keep the load balanced with its neighbors; iteratively, the loads in a system will be evenly distributed among the nodes. We conjectured that a diffusive scheme would well manage the resources in a P2P system.

Within our proposed diffusive load balancing scheme, each cluster runs the load balancing procedure described in subsection B. In our simulations, each cluster initiates the procedure after a regular, partially random, time interval. It is assumed that the time required to complete the procedure is short compared with this time interval. There is no coordination between different clusters for initiating the procedure.

For our simulations, we assume a clustered P2P system similar to eQuus [1]; each node could join any cluster. We assume that each cluster has a designated decision node that performs the load balancing procedure for the cluster. When such a node leaves due to churn, the other nodes of the cluster will determine a new decision node.

A. Load Index: available capacity

Any load balancing procedure uses some **load index** which is the measure of the load status that should be equalized throughout the system. Some authors have used the mean response time of requests for load balancing in client/server system [13]. There is a direct relation between the server's available capacity and the mean response time as we discuss here.

We assume that the performance of each node can be described by a queuing model. From the M/M/1 queuing model and Little's Law, we derive a direct relationship between the node's mean response time E[r] and its available capacity: E[r] = 1/available_capacity [15]. This indicates that if two servers have the same available capacity, the mean response time of their services are the

same. We conclude that we get a uniform response time if our load balancing procedure uses as load index for a cluster the average available capacity of the nodes in that cluster.

In a clustered P2P system, we assume that the load of the nodes in a given cluster has already been balanced, hence, these nodes have the same available capacity. However, the available capacities of nodes in different clusters are not the same. Our load balancing scheme works at the inter-cluster level to equalize the available capacities of nodes in different clusters.

B. The load balancing procedures

As we indicated before, a node in a cluster will be selected as decision node and periodically run the load balancing procedure. Next, we describe this procedure with its four phases.

Triggering Phase: In this phase, a decision node invokes a new round of the balancing procedure starting with the load determination phase. The event that triggers the invocation could be a timeout event, or the observation that the load index of the cluster has reached a static threshold. Currently, we only consider the first case.

Load Determination Phase: The cluster determines its own load status as well as the load status of its neighborhood. The neighborhood of a given cluster is all those clusters that are contained in the DHT routing table of that cluster. The decision node sends probing messages to these neighbors, and waits for their responses; a probed cluster responds with its load index. The average_load_index (of the neighborhood) will be calculated according to these responses.

Decision Phase: A dynamic threshold is used to determine whether a cluster is considered overloaded or under-loaded. The upper and lower thresholds are calculated by using the formula: threshold = average_load_index * (1 +/- bound). The bound is given in terms of the percentage of the average load index of the neighborhood. The detail of the decision procedure depends on the *Location* policy, which determines the pair (or pairs) of sender (overloaded) and receiver (underloaded) for a load transfer:

Directory-initiated: the decision node identifies one or several receiver-sender pairs in its neighborhood. The senders are overloaded clusters, and the receivers are clusters either underloaded or regularly loaded.

Sender-initiated: if the cluster of the decision node is a sender (over-loaded), then it tries to identify a corresponding receiver in its neighborhood.

Receiver-initiated: If the cluster of the decision node is a receiver (under-loaded), then it tries to identify a corresponding sender in its neighborhood.

Load transfer Phase: The decision node will send a load transfer request to the receiver of each identified sender-receive pair. When a receiver cluster receives a load transfer request, it will select one of its nodes, delete it from its membership list, and let it join the sender cluster. It is important that the node movement should not cause the state of these clusters to be changed to the opposite, e.g., an under-loaded cluster should not become overloaded, or, an overloaded cluster should not become under-loaded. In order to avoid such situations, a receiver can only transfer out the portion which is above the mean of its neighborhood.

V. PERFORMANCE ANALYSIS

We investigate the proposed load balancing scheme through simulation. The balancing procedure run in a simulator with an overlay network which conducts the operations of a clustered DHT, including churn through node joining and leaving which lead to clusters splitting and merging, and the resulting routing table updates. We implemented the three location policies described above and, for comparison, a central directory scheme performed by a central directory with global knowledge.

We evaluate load balancing procedures according to two aspects: (1) balancing result, and (2) balancing behavior. For the first aspect, we look at the standard deviation of the load indexes of clusters reached by the load balancing procedure, and the distance between the minimum load index of a cluster and the average load index of the clusters in the system, which we call the *delta*. For the second aspect, we look at the impact of load balancing on the system, including the number of node movements for the purpose of load balancing and the number of splits or mergers of clusters occurring in the system. An effective scheme should lead to evenly distributed loads among the clusters without introducing too many node movements and cluster splits and mergers.

TABLE I. LIST OF PARAMETERS OF THE SIMULATION

Interval of two consecutive runs of the central directory load balancing procedures	Uniformly distributed in the range 1000 +/- 20% seconds
Interval of two consecutive runs of the load balancing procedures for a given cluster	Uniformly distributed in the range 1000 +/-5% seconds
Total number of nodes	10000
Homogeneous node capacity	10
Heterogeneous node capacity	Pareto distribution [100, 5000], shape = 2, scale = 100 [17]
Number of nodes in a cluster	[4, 16]
Base used in the digits of the cluster IDs	4
Churn per load balancing period : k	[0, 100]
Frequency of system status measurements	Every 1000 seconds

The input parameters for the simulation are listed in Table I. We assume that the system is in a steady state where the processes of nodes joining and leaving are Poisson processes with the same rate. While churn is defined as the ratio of the state changes per time unit over the total number of nodes in the system [16], we use a parameter k defined by k = churn * average load balancing period, called *normalized churn*, which

represents the fraction of nodes that have joined or left, on average, during a load balancing period of a cluster. For example, when k is 2%, there are 1% of the nodes that have left and 1% that have joined when the load balancing procedure runs again at the end of one load balancing period.

We investigate load balancing for system with homogeneous and heterogeneous node capacities. For the homogeneous system, all nodes have the same capacity, and different *location* policies are compared. For heterogeneous systems, Pareto distribution is used to represent the node capacities, and different selection policies are compared.

A. Load balancing in a system with churn and homogeneous node capacities

We have first performed simulations of a P2P system with low churn. At the beginning of the simulation, the load for the different clusters is selected such that the load index of the clusters follows a uniform distribution between zero and the maximum capacity. This load, for each cluster, remains fixed throughout the simulation; we note, however, that when a cluster is split, the load is shared half-and-half by the two resulting clusters, and when two clusters merge, the resulting cluster has a load corresponding to the sum of the two joining clusters.

After a simulation run of 50 measurement periods with light churn of k = 5%, and no load balancing, we observe that the distribution of the load index has a long left tail in a range less than 0 (this means that some clusters are completely overloaded); in fact, about10% of the clusters have negative available capacities. This means that the capacity required for processing the requests exceeds the capacity of the nodes in these clusters.

TABLE II. COMPARISON OF DIFFERENT LOCATION POLICIES

	splits	**mergers**	**node moves**	**std. dev**	**delta**
CD	8.42	3.96	102.8	0.71	2.54
DI	7.67	3.63	123.5	0.55	1.46
SI	8.06	4.23	102	0.83	3.22
RI	7.58	3.75	254.1	0.95	2.09

Then we have done simulations with load balancing until a stable system state is reached. The results obtained in the stable states for the different location policies are shown in Table II. The directory-initiated version (DI) has better values for the standard deviation and delta of the available capacities than the sender-initiated (SI) and receiver-initiated (RI) versions; however, it causes more node movements than the central directory (CD) and send-initiated versions. The results also indicate that sender-initiated location policy can not fully remediate all the overloaded clusters.

It is important for the load balancing algorithm to capture the load dynamics in a system. We now show how the directory-initiated algorithm performs under different churn rates; we compare it with the central directory scheme in Table III. The imbalance factor: *delta* (%) is the difference between the average and the minimum load among all clusters in the system, normalized to the average.

TABLE III. COMPARISON OF CENTRAL DIRECTORY SCHEME (CD) AND DIRECTORY-INITIATED SCHEME (DI) UNDER DIFFERENT CHURN RATES

churn rate%		**20**	**40**	**60**	**80**	**100**
split%	**CD**	1.13	1.66	2.56	3.697	4.56
	DI	0.26	0.24	0.48	0.788	1.07
merge%	**CD**	0.59	1.12	1.97	3.421	4.19
	DI	0.05	0.16	0.53	0.741	0.88
node mv%	**CD**	2.53	3.92	4.47	6.111	6.53
	DI	3.84	8.08	11.7	14.03	15.1
std dev.	**CD**	0.93	1.11	1.22	1.369	1.63
	DI	0.52	0.54	0.58	0.604	0.64
delta/mean %	**CD**	86.2	111	115	142.4	183
	DI	44	58.1	63	64.66	70.3

In a system with churn, the directory-initiated policy always performs better than the central directory scheme. The central directory version used in our simulation performs at most one node exchange between a single sender-receiver pair per measurement period. When the churn increases, it cannot perform enough node movements to counterbalance the effect of churn. On the other hand, the directory-initiated version runs the load balancing algorithm on average on all clusters during one measurement period, and each cluster may initiate a node movement. It is therefore much more responsive than our central directory.

Both versions have the *delta* increasing with growing churn rate. This indicates that a small number of clusters have not been balanced, while a large portion of clusters have their load indexes close to the average of the system. The increase of the *delta* is much slower for the directory-initiated version.

To further improve the balancing results of the directory-initiated version, a smaller load balancing period could be used, or the sender-initiated scheme could be started in a cluster when the cluster has an outstanding capacity shortage. We will investigate these possibilities in our future research.

Both schemes add extra node movements to the system. However, we observe that, compared with a system without balancing, the number of splits and merges in the system with a balancing scheme are reduced, especially, with the directory-initiated scheme. When the churn rate is 100%, in an unbalanced system, there are 13.8% clusters splitting and 14.0% clusters merging; with directory-initiated scheme, there are only 1.07% clusters

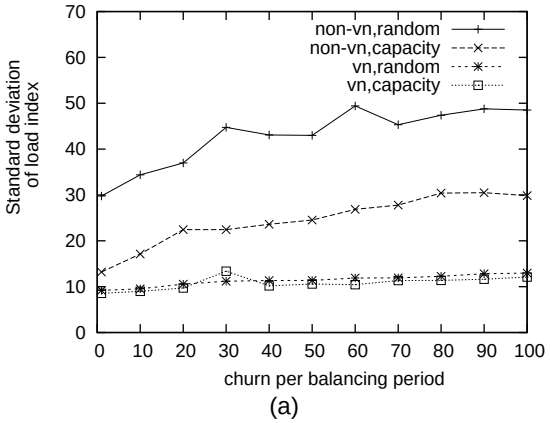
splitting and 0.88% clusters merging in the system. This indicates that a balancing scheme could maintain the cluster organization through moving nodes around; this increases the stability of the clustering structure.

B. Maintaining the stable state in a system with heterogeneous node capacities

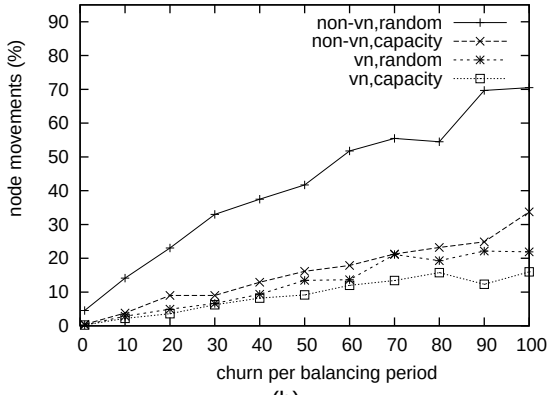
In a system with heterogeneous node capacity, we compare in the following two selection policies: (a) the *random selection policy* which selects a random node from the receiver cluster for transfer to the overloaded cluster, and (b) a *capacity consideration policy* that tries to select a node with a maximum capacity just sufficient to increase the load index of the overloaded cluster to the perceived mean.

We use a Pareto node capacity distribution (as described in Table I) for the following experiment. We introduce virtual nodes into the heterogeneous system: when the capacity of a node is above a certain boundary, its capacity will be divided into several virtual nodes, and these virtual nodes will be inserted as members into clusters that are randomly selected from the clusters in the system. When such a high-capacity node leaves the system because of churn, all of its virtual nodes will leave their respective clusters. The impact of this leaving is therefore distributed over many clusters and induces much less load imbalance. For the simulation, we selected 400 as the upper capacity boundary, which would be exceeded by 6.25% of the nodes, and each of these nodes would be divided into virtual nodes with a capacity of 200.

Figure 1 compares directory-initiated load balancing with and without virtual nodes. Without virtual nodes, the capacity consideration policy is superior to the random selection policy (Figure 1(a)); however, the load balancing performance is not satisfactory: the standard deviation of the load index of the nodes is still large. After introducing virtual nodes, the standard deviation of the load index of the nodes becomes smaller.



(a)



(b)

non-vn: without virtual node, vn: with virtual node,
random: random selection policy,
capacity: capacity consideration policy.

Figure 1. Directory-initiated load balancing with capacity consideration selection policy under various churn rates: (a) standard deviation of load index, (b) node movements

Compared to the random selection policy, the capacity consideration policy causes less node movements (Figure 1(b)). When the system uses virtual nodes with capacity consideration policy, the amount of node movements is close to in the case of a homogeneous system (Table III); we also observe the same trend in the amount of splits and merges. The advantage of the capacity consideration policy is clear when the nodes have heterogeneous capacity.

This experiment shows that using virtual nodes on extreme high capacity nodes can largely improve the performance of load balancing.

VI. CONCLUSION

We designed a diffusive load balancing algorithm for a P2P system with node clustering. The algorithm balances the available capacity of all clusters in the system. Since the available capacity of nodes is directly related to the response time provided to similar requests, the load balancing algorithm leads to a uniform response time for all nodes in the system.

This paper investigates the directory-initiated balancing scheme in systems with nodes of homogeneous or heterogeneous capacity. When the system experiences churn, the balancing algorithm keeps the available capacities of the nodes close to their average. In heterogeneous systems, when nodes have highly diverse capacities, a proposed capacity consideration selection policy is superior to random node selection; however, when the system uses virtual nodes for extremely high capacity nodes, the difference between these two selection policies becomes small.

Generally, load balancing induces extra node movements into the system. However, the number of cluster splits and merges is reduced; this reduction decreases the amount of changes in the routing table. This

means that balancing improves the stability of the clustered P2P system.

In our future research, we plan to explore the balancing algorithm under various load distributions, and also consider dynamic loads with non-stationary arrivals. In addition, we will try to find schemes for further reducing the overhead introduced by load balancing and the associated node movements.

REFERENCES

[1] T. Locher, S. Schmid, R. Wattenhofer, "eQuus: A Provably Robust and Locality-Aware Peer-to-Peer System," In *Proceeding of Sixth IEEE International Conference on Peer-to-Peer Computing (P2P'06)*, 2006, pp. 3-11.

[2] S. Surana, B. Godfrey, K. Lakshminarayanan, R. Karp, and I. Stoica, "Load balancing in dynamic structured peer-to-peer systems," in *Performance Evaluation Volume 63, Issue 3, P2P Computing Systems*, March 2006, Pages 217-240.

[3] F. Kuhn, S. Schmid, R. Wattenhofer, "A Self-repairing Peer-to-Peer System Resilient to Dynamic Adversarial Churn," In *Proceedings of the 4th IEEE International Workshop on Peer-to-Peer Systems (IPTPS)*, Cornell University, Ithaca, New York, USA., February, 2005.

[4] Bertsekas, D. P. and Tsitsiklis, J. N. *Parallel and Distributed Computation: Numerical Methods*. Athena Scientific, 1997

[5] V.A., Saletore, "A Distributed and Adaptive Dynamic Load Balancing Scheme for Parallel Processing of Medium-Grain Tasks," *Distributed Memory Computing Conference, 1990, Proceedings of the Fifth* , vol.2, no., pp.994-999, 8-12 Apr 1990.

[6] J. Byers, J. Considine, and M. Mitzenmacher. "Simple Load Balancing for Distributed Hash Tables," In Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS '03), February 2003.

[7] Y. Zhu, Y. Hu. "Efficient, proximity-aware load balancing for DHT-based P2P systems," *IEEE Transactions on Parallel and Distributed Systems*, vol.16, no.4, pages 349-361, April 2005

[8] A. R. Bharambe, M. Agrawal, and S. Seshan, "Mercury: supporting scalable multi-attribute range queries," In *Proceedings of the 2004 Conference on Applications, Technologies, Architectures, and Protocols For Computer Communication*. SIGCOMM '04. ACM, New York, NY.

[9] P. Ganesan, B. Mayank, and H. Garcia-Molina. "Online Balancing of Range-Partitioned Data with Applications to Peer-to-Peer Systems," *in VLDB, 2004*.

[10] G. Cybenko, Dynamic load balancing for distributed memory multiprocessors. *J. Parallel Distrib. Comput.* 7, 2 (Oct. 1989), pages 279-301

[11] B. Monien and R. Preis, Diffusion schemes for load balancing on heterogeneous networks, *Theory of Computing Systems, vol 35*, 2002.

[12] A. Corradi, L. Leonardi, F. Zambonelli, Diffusive Load-Balancing Policies for Dynamic Applications, *IEEE Concurrency, vol. 7, no. 1*, pp. 22-31, Jan.-Mar. 1999,

[13] M.-V. Mohamed-Salem, G. v. Bochmann, and J. W. Wong, "Wide-area server selection using a multi-broker architecture," in *Proceedings of International Workshop on New Advances of Web Server and Proxy Technologies.* Providence, USA, May 19, 2003.

[14] A. Cortes, A. Ripoll, F. Cedo, M. A. Senar, E. Luque, "An asynchronous and iterative load balancing algorithm for discrete load model," Journal of Parallel and Distributed Computing, Volume 62, Issue 12, December 2002, Pages 1729-1746.

[15] Y. Qiao, G. v. Bochmann, "Applying a diffusive load balancing in a clustered P2P system." in Proceeding of 9th International conference on New Technologies of Distributed Systems (NOTERE), Montreal, Canada, 2009.

[16] P. B. Godfrey, S. Shenker, and I. Stoica, "Minimizing churn in distributed systems," in *Proceedings of the 2006 Conference on Applications, Technologies, Architectures, and Protocols For Computer Communication*, SIGCOMM '06. Pisa, Italy, September 11 - 15, 2006

[17] B. Godfrey, K. Lakshminarayanan, S. Surana, R. Karp, I. Stoica, "Load balancing in dynamic structured P2P systems," *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, vol.4, page. 2253-2262 7-11 March 2004.