A PHYSICS-INSPIRED PERFORMANCE EVALUATION OF A STRUCTURED PEER-TO-PEER OVERLAY NETWORK

Sameh El-Ansary
Swedish Institute of
Computer Science (SICS)
Sweden
email: sameh@sics.se

Erik Aurell
Department of Physics, KTH
Royal Institute of Technology
Sweden
email: eaurell@sics.se

Seif Haridi
Department of Physics, KTH
Royal Institute of Technology
Sweden
email: eaurell@sics.se

ABSTRACT
In the majority of structured peer-to-peer overlay networks a graph with a desirable topology is constructed. In most cases, the graph is maintained by a periodic activity performed by each node in the graph to preserve the desirable structure in face of the continuous change of the set of nodes. The interaction of the autonomous periodic activities of the nodes renders the performance analysis of such systems complex and simulation of scales of interest can be prohibitive. Physicists, however, are accustomed to dealing with scale by characterizing a system using intensive variables, i.e. variables that are size independent. The approach has proved its usefulness when applied to satisfiability theory. This work is the first attempt to apply it in the area of distributed systems. The contribution of this paper is two-fold. First, we describe a methodology to be used for analyzing the performance of large scale distributed systems. Second, we show how we applied the methodology to find an intensive variable that describe the characteristic behavior of the Chord overlay network, namely, the ratio of the magnitude of perturbation of the network (joins/failures) to the magnitude of periodic stabilization of the network.

KEY WORDS
Peer-to-Peer overlays, DHT performance, Structured Overlay networks, Data Collapse, Complex Systems

1 Introduction

A number of structured P2P overlays [17, 14, 18, 8, 2], aka Distributed Hash Tables (DHTs) were recently suggested. In most such systems, nodes *self-organize* in a graph with a diameter and outgoing arity of nodes that are both of a logarithmic order of the number of nodes. Some systems like [7, 13] can even provide a logarithmic order diameter with a constant order routing table. To maintain the graph in an optimal state despite change of membership (joins/failures), each node needs to follow some *self-repair* policy to keep its routing table (the outgoing edges) up-to-date.

Our general observation on the literature of structured overlay networks there is more work on the self-organization aspect compared to the self-repair aspect due to the following two factors: $i)$ The novelty of such systems and the requirement to establish the new concepts first before deeper analysis is performed. $ii)$ Once we have a system deploying a self-repair policy, an analytical model that describes the behavior of the system can range in degree of difficultly from not-so-clear to very-complex-to-analyze. Therefore, we see the compelling need for more studies whether analytical or simulation-based that can tell us whether those novel techniques are really useful or over-hyped.

In this paper we try a novel approach for analyzing the performance of one of the most well-established overlay networks, namely the Chord system [15, 16, 17]. In the following section we motivate our adoption of a physics inspired approach and we state our methodology. After that, we explain our assumptions in implementing the Chord system. We follow that by the core sections of the paper namely the application of our methodology to find intensive variables. Finally, we conclude the work in the last section.

2 The Physics-Inspired Approach

2.1 Motivation.

Having observed that analytical models are not always trivial to formulate given a system applying a given self-repair policy, we thought that using simulation seemed to be the practical tool for analyzing such systems. However, we figured out that scales of interest could be prohibitive for simulation purposes. At this point, a physics-style approach started to be of interest since physicists are accustomed to reasoning about large natural systems.

2.2 How Do Physicists Deal with Scale?

Physics was the first science to encounter problems of this sort. The number N of molecules in a macroscopic body, say a liter of water, is about 10^{27}. On the microscopic level, all substances are made of atoms and molecules of the same basic type – different number of electrons, protons and neutrons – yet large lumps of the same kind of atoms or molecules make up substances with distinct qualities which we can perceive. Those can be density, pressure (of a gas at given density and temperature), viscosity (of a liquid), conductivity (of a metal), hardness (of a solid), how sound and light do or do not propagate, if the material is magnetic, and so on.

The first level of analysis in a physical system of many components, is to try to separate *intensive* and *extensive* variables. Extensive variables are those that eventually become proportional to the size of the system, such as total energy. Intensive variables, such as density, temperature and pressure, on the other hand, become independent of system size. A description in terms of intensive variables only is a great step forward, as it holds regardless of the size of the system, if sufficiently large.

Further steps in a physics-style analysis may include identifying phases, in each of which all intensive variables vary smoothly, and where the characteristics of the system remain quantitatively the same.

2.3 Was the Approach Useful in the Computer Science Arena?

A physics-style approach was carried over to satisfiability theory more than ten years ago. KSAT is the problem to determine if a conjunction of M clauses, each one a disjunction of K literals out of N variables can be satisfied. Both M and N are extensive variables, while $\alpha = M/N$, the average number of clauses per variable, is an intensive variable. For large N, instances of KSAT fall into either the SAT or the UNSAT phase depending on whether α is larger or smaller than a threshold $\alpha_c(K)$ [10, 3]. The order of the phase transition, a statistical mechanics concept roughly describing how abrupt the transition is, has been shown to be closely related to the average computational complexity of large instances of KSAT with given values of K and α [12, 11]. Recent advances include the introduction of techniques borrowed from the physics of disordered systems, leading to an important new class of algorithms, currently by far the best of large and hard SAT problems [9]. Without question, statistical mechanics have been proven to be very useful on very challenging problems in theoretical computer science, and it can be hoped that this will also be the case in the analysis and design of distributed systems.

2.4 "Data Collapse": the Tool for Observing Intensive Variables

Let us assume that we have a system that we evaluate using a function $m(S, P)$ where m is some performance metric, S is the size of the system and P is a set of system parameters. A description of the system in terms of all the possible values of m under all values of S and P is not very transparent, and can be prohibitive to enumerate.

If all the parameters in P are essential, we cannot do better. If on the other hand they only influence m through some combination or functional relationship, it is preferable to use instead $m(g(\psi))$, where $\psi \subset \{S\} \cup P$, and g is some function. We will assume g smooth, as otherwise the practical utility is generally small. The main advantage is that this encodes a definite understanding of what really influences system behavior. Additionally, the data can be presented in a systematic and much compact manner.

A discussion of the process of obtaining ψ and g in general is outside this presentation. Suffice it so say that if the relationship is simple, say linear, it can be found by exhaustive search. Once however a putative relationship has been posited, it can be tested for by systematically varying all parameters. If there is indeed a *functional relationship*, the relation $(g(\psi), m)$ has to be injective. In a diagram of m versus g, all data points must then fall on one curve. This phenomenon, if it occurs, is called "data-collapse", and serves to prove that the posited relationship correctly describes the system. If we were to apply that to the K-SAT case mentioned in section 2.3, m is the fraction of unsatisfiable instances, and $\psi = \{M, N\}$ and $\alpha = g = \frac{M}{N}$.

2.5 Application of the Approach in Distributed Systems

Having explained the importance of identifying intensive variables in describing characteristics that hold irrespective of size and shown the pragmatic tool to find such variables, namely, data collapse, we state the main question that we try to answer in this work and summarize the methodology we use to answer that question:

> *"Is it possible to find intensive variables to describe the characteristics of structured peer-to-peer overlay networks that deploy periodic maintenance of the overlay graph?"*

The Methodology

Step 1: Nomination of intensive variables. This step is a speculative step where we try to nominate a subset of the parameters ($\psi \subset \{S\} \cup P$) that affect the performance and speculate that a function of them ($g(\psi)$) can be an intensive variable.

Step 2: Looking for a performance metric. In this step a performance metric $m(g(\psi))$ is identified and is usually an easier step for systems where a performance metric is already agreed upon in the community but the step can also involve the introduction of new metrics.

Step 3: Simulation. With plotting m versus g in mind, we chose a range of values of ψ to explore a wide spectrum of parameter interaction. Finally, if a data collapse is observed, then an intensive variable is identified.

3 Background & assumptions about Chord

Despite the fact that the Chord system is one of the most clearly explained DHT systems, when implementing it there are some nuances in the implementation details that can affect the performance of the system. We provide here a semi-formal definition of Chord and state our assumptions when needed.

- **The Chord graph**: $G = (V, E)$ where V is the set of nodes (machines/processes) and E is a set of directed edges.
- **Identifier** Id_u: Each node in the set V has a unique identifier obtained by hashing a unique property such as its IP address or public key. We write Id_u to refer to the Id of node $u \in V$ and we label the edges using the corresponding identifiers of the nodes.
- **Size of the identifier space** N: The number of possible identifiers. The Chord system assumes that all the nodes are totally ordered in a circle that has N positions.
- **Population size** P: The number of nodes $P = |V|$, $1 \leq |V| \leq N$.
- **Successor of node** u**,** $Succ(Id_u)$: The successor of an identifier Id_u is the first node following u in the circular identifier space. e.g if $N = 16$, $V = \{3, 11\}$, then we have two nodes whose identifiers are 3 and 11. Therefore the successor of any of the identifiers $4, 5, ...10, 11$ is node 11, similarly the successor of any of the nodes $12, 13, 14, 15, 1, 2, 3$ is node 3.
- **Routing table of node** u**:** $RT(Id_u) = \{(Id_u, Id_v) : Id_v = Succ(Id_u + 2^{i-1})\}, (1 \leq i \leq \log_2 N)$, where addition it modulo 2.
- **Succesor List of node** u**:** $SL(Id_u) = \{(Id_u, Id_v) : Id_v = Succ(Id_u)$, $Succ(Succ(Id_u))$, $Succ(Succ(Succ(Id_u)))$, ... upto $\log_2 N$ successors $\}$
- **Lookup process:** A lookup of Id_q at a node u results in u forwarding the lookup to a node v pointed to by the highest edge of u preceding Id_q. v acts similarly. With each crossing of an edge (hop), the distance to Id_q decreases by a factor of a half at least and thus after $O(\log_2 P)$ hops the lookup reaches its goal.
- **Joins:** When a new node joins the system through any other node, it performs a lookup to find out its successor and then uses a periodic stabilization algorithm to correct all its edges. During that process, both its edges, and the edges of other nodes that should point to it are out-of-date and this results in lookups taking more hops to be resolved.
- **Failures:** When a node fails ungracefully - that is, without notifying the nodes pointing to it of its failure - lookups from nodes who have edges pointing to the failed node will fail and will be retried with lower edges. This also results in the lookups taking more hops to be resolved.
- **Stabilization:** There has been a couple of stabilization algorithms suggested for reparing the routing table. One iterates through the edges in a round-robin fashion and another picks randomly one of the edges. In our implementation we use the second. For the stabilization of the successor list, there was a non-formal description in the Chord paper, and we adopted a naive interpretation where each nodes asks its successor for its successor list and blindly replaces its own dropping the last entry. We also assume for simplicity that the stabilization period for the routing table and the successor list is the same.

4 Ratio of Perturbation to Stabilization (β) as an Intensive Variable

We like to perceive a structured overlay network as a system operating under two competing forces: perturbation and stabilization. Perturbation is the change in the set of nodes by adding and removing nodes, aka "churn". Stabilization is the periodic maintenance of edges performed by each node. Perturbation pulls the network towards sub-optimal performance because whenever a node is added or removed, some outgoing edges of some other nodes need to be updated. Stabilization brings the network back to optimal state by reassigning sub-optimally-assigned and stale edges. The competition between those two forces governs the performance of the system. Therefore, the question we are investigation is:

> **Question** 1**.** *If a system of size S is operating under a magnitude of perturbation x and a magnitude of stabilization y, what is the performance of the system according to some performance metric $m(x, y)$?*

A comprehensive simulation-based answer to question 1 can not be obtained without an exhaustive exploration of all the parameters which is prohibitive. Instead, we try to answer it by posing the two following simpler questions which in essence are attempts to find data collapse:

> **Question** 2 **(Perturbation-Stabilization Equilibrium).** *Assume that we know $m(x_1, y_1)$ where x_1 and y_1 are some values of perturbation and stabilization respectively, is $m(x_1, y_1) = m(\alpha \times x_1, \alpha \times y_1)$ where α is a constant? Differently said, is there an equilibrium of those two competing forces?*

> **Question** 3 **(Scalability of the equilibrium).** *Whatever the answer to question 2 is, how does it hold for larger system sizes?*

4.1 Application of the Methodology

To answer the above two questions we apply our methodology as follows:

Step 1: Nomination of intensive variables. Let τ be the time between two stabilization actions of a certain node. Let μ be the average time between two perturbation events (joins or failures) while the network is in a stable state. That is, the number of nodes is varying around a certain average population P_0. Taking μ as the magnitude of perturbation and τ as the magnitude of stabilization, we need to answer the following question: "Is $\beta = \frac{\mu}{\tau}$ an intensive variable?".

Step 2: Looking for characteristic behavior. In this investigation, the average lookup path length is the indicator of characteristic behavior. In addition to that, we need a metric that gives more insight about the state of the graph and thus we use what we call the "distance from optimal network" δ which is computed as follows:

$$\delta = \frac{\sum_{i \in P} \sum_{j=1..\log N} \text{Edge}_j^i \neq \text{OptimalEdge}_j^i}{P \log N} \qquad (1)$$

Where Edge_j^i is the $j^{th}(1 \leq j \leq \log N)$ outgoing edge of a node $i \in P$ and OptimalEdge_j^i is the optimal value for that edge. $P \log N$ is the total number of edges (P nodes, $\log N$ edges per node, where N is the size of the identifier space). Informally, δ is the number of "wrong" (outdated) edges in the overlay graph over the total number of edges. It varies between 0 and 1 where a value of 0 means that the graph is optimal.

Step 3: Simulation Setup.
We let P_0 nodes form a network and we wait until δ is equal to 0 i.e. the network graph is optimal. We then let the network operate under specified values of μ and τ for 50 turnovers (A turnover is the replacement of P nodes with another P). During this experiment, we record δ frequently and average it over the whole experiment. That is in addition to the average lookup path length L over the whole experiment as well. We write $\langle\delta(P, \tau, \mu)\rangle$ and $\langle L(P, \tau, \mu)\rangle$ to denote the average δ and L obtained by running this experiment setup under given values of P, τ and μ.

Figure 1 shows example experiments. On the x-axis two values are plotted at each discrete time t: $i)$ δ at time t, $ii)$ $\frac{P_t}{P_0}$, the population at time t divided by the initial population P_0. During the experiment, the average population is P_0 (That is $\left\langle \frac{P_t}{P_0} \right\rangle = 1.0$). In figure 1(a) the values of τ and μ were 150 and 30 respectively, therefore $\beta = 0.2$. In figure 1(b) both values of τ and μ are doubled while keeping beta constant. Interestingly, the average distance in both examples is almost the same, that is $\langle\delta(128, 150, 30)\rangle = 0.518$ and $\langle\delta(128, 300, 60)\rangle = 0.504$. While those examples are promising for showing that an equilibrium of perturbation and stabilization exists, we show the investigation of a broad range of values of β in the next section.

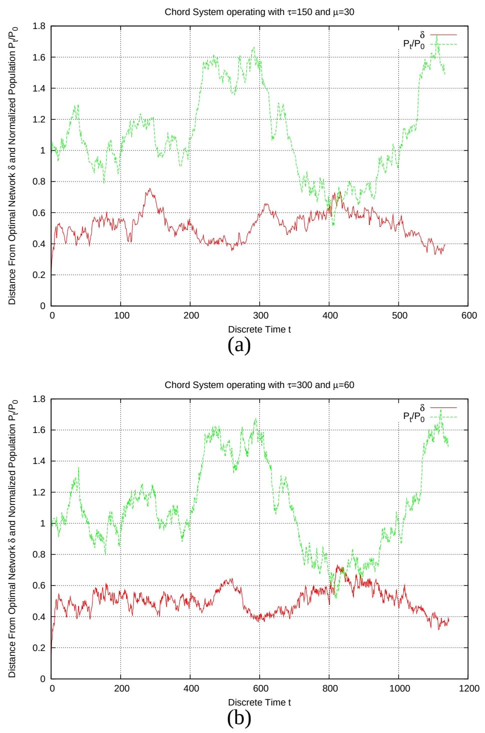

Figure 1. Example experiments showing the average normalized population size of 128 nodes under perturbation (joins/failure) and the average distance from optimal network under two different rates of perturbation (μ) and stabilization (τ) but the same $\beta = \frac{\mu}{\tau}$

4.2 Results

4.2.1 Perturbation-Stabilization Equilibrium

For one value of P_0, we examine various values of β by fixing τ and varying μ. We, then, try a different τ and vary μ such that the same values of β are conserved. The target of this procedure is to understand whether δ is dependent solely on β or is it one of the components of β (τ or μ) that controls $\langle\delta\rangle$ and $\langle L\rangle$.

In more details, we try values of $\tau = \{150, 300, 600, 1200\}$. For each value of τ we try different values of μ. For instance, for $\tau = 150$, values of $\beta = \{25, 30, 45, ..., 195, 200\}$. For $\tau = 150$, values of $\beta = \{50, 60, 70, ..., 390, 400\}$. The same procedure is repeated for values of $P_0 = \{64, 128, 512, 1024\}$.

The results are shown in figures 3 and 2. Observe that as β increases the time between two perturbation events increase, i.e. the network has more chance to heal itself. Therefore, it is expected to see that both $\langle L\rangle$ and $\langle\delta\rangle$ decrease indicating a better performance.

In figure 3 we find that for smaller network sizes, the

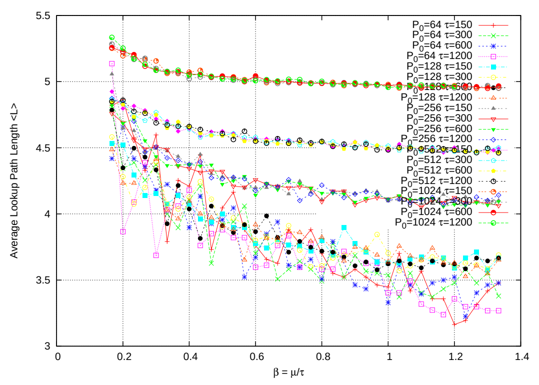

Figure 2. The average lookup path length $\langle L \rangle$ as a function of the speculated intensive variable β (the ratio of average time between perturbation events μ and average time between stabilization events τ).

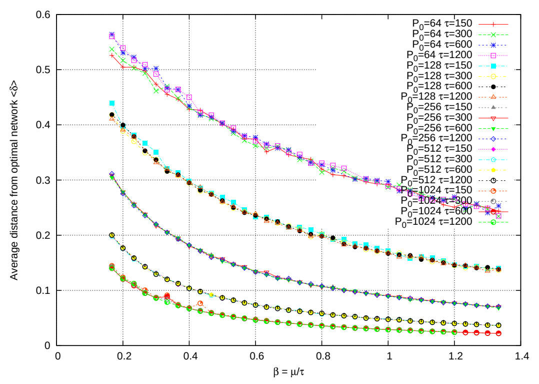

Figure 3. The average distance from optimal network $\langle \delta \rangle$ as a function of the speculated intensive variable β.

average lookup length oscillates strongly and it is not clear that β alone controls the behavior. For larger sizes, this oscillation disappears and we clearly see that the curves superimpose nicely and a data collapse is obtained. Another way to see the behavior of the lookup length is to plot its deviation from the optimal value ($\langle L_{opt} \rangle = 0.5 \times \log \langle P_0 \rangle$) as shown in figure 4.

Unlike the average lookup path length, the distance from optimal network gives a much more consistent view of the network behavior. As shown in figure 3, for each given size all the curves superimpose nicely. That is, we have a much more vivid data collapse. A discussion of a data collapse for all the sizes is provided in the next section. However, at this point, we can see that the network exhibits a perturbation-stabilization equilibrium.

4.2.2 Scalability of the equilibrium

If we want to discuss the scalability of the equilibrium, we need to perceive the obtained results differently. The stabilization as defined in section 4.1 is a "node-level" event

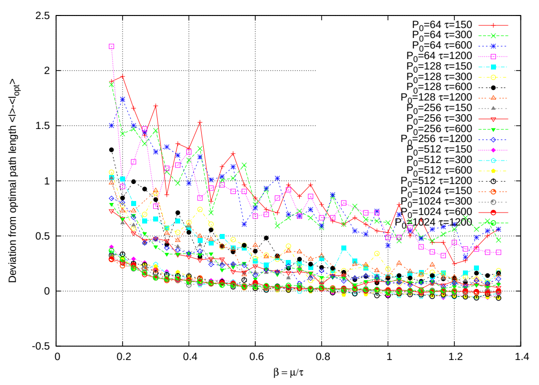

Figure 4. The deviation from optimal average lookup path length $\langle L \rangle - \langle L_{opt} \rangle$ as a function of the speculated intensive variable β.

while the perturbation is a "whole-graph-level" event. That is, β is defined as $\frac{Perturbation\ of\ the\ system\ (\mu)}{Stabilization\ of\ each\ node\ (\tau)}$. Therefore, if we were to compare the behavior of two network sizes under the same values of τ and μ, the network with larger size will have the same perturbation but higher stabilization since the number of nodes is larger. Therefore, we define β' to be $\frac{Perturbation\ of\ the\ system\ (\mu)}{Stabilization\ of\ the\ system\ (\frac{\tau}{\langle P_0 \rangle})}$ to have a more fair comparison. The β' re-plots of figure 4 and 3 are shown in figures 5 and 6 respectively.

In figure 5 we can see that we have a noisy data-collapse due to the oscillations of the lookup length of the small networks. Nevertheless, a clear common trend is obvious and we believe that enhancing it more is an exercise in statistical methods. However, judging it by using the distance from optimal network (figure 6), we can see that we have a clear data collapse. With that, we can conclude that the ratio of system perturbation to system stabilization β' in a Chord system is an intensive variable.

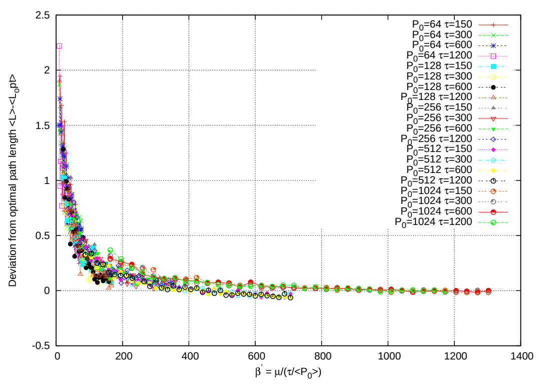

Figure 5. Data collapse of figure 4 obtained by using β'.

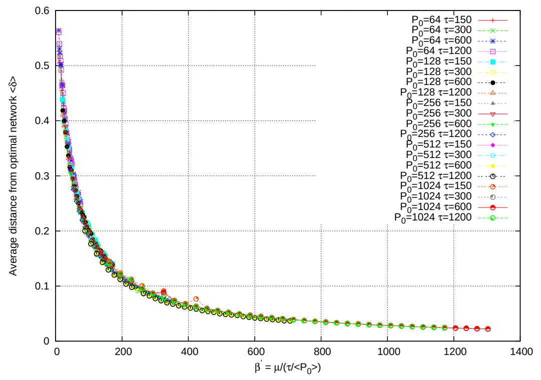
Figure 6. Data collapse of figure 3 obtained by using β'.

5 Related Work

We consider in this section the research in the structured overlay networks community, that we are aware of, which addressed self-repair on deep levels.

A lower bound on stabilization rate. In [5], the Chord research team gives the first theoretical analysis on the lower bound of stabilization rate required for a network to remain connected in face of continuous joins and failures. A lower bound for the stabilization rate is definitely a substantial result, however, we are interested to answer more questions regarding the stabilization rates such as: If we consider the range of stabilization rates where the overlay graph won't be disconnected, the guarantee of graph disconnection is not sufficient for practical purposes, how is the network performance affected by each such rate? Differently said, if we have an application that demands a limit on the latency in terms of the number of hops, what is the required stabilization rate?

Self-tuned message failure rate. In [6], the Pastry research team gives an analytical model for the percentage of message loss due to outdated routing tables as a function of the frequency of routing table probes. The model is then used to adaptively "self-tune" the rate at which the network self-repairs its routing entries that are pointing to failed nodes which is another major result because of its practical implications. However, the analysis based on the failed nodes only is not sufficient since a node can have alive but sub-optimally assigned edges.

Comparing performance under churn. The work in [4] gives the first thorough simulation-based analysis for overlays network under churn (perturbation). In that sense it is the most similar research to our work. The simulation model covers physical network proximity and compares several systems and not only Chord. On the other hand, that simulation is based on one churn rate for one system size while in ours we tackle several rates and we are interested in finding data-collapse so as to present results that hold irrespective of the network size.

6 Note on the implementation.

To perform the experiments, we used the Mozart [1] distributed programming platform to implement a discrete event simulator. We wrapped our simulator in a distribution layer to be able to schedule experiments on a cluster of 16 nodes at SICS. Each nodes is an AMD Athlon(tm) XP 1900+ (1.5GHz) with a 512MB of memory.

7 Conclusion and future work

We have reported in this paper our progress in investigating whether a physics-style analytical approach can give more understanding to the performance of structured overlay networks. The approach mainly necessitates the description of the characteristics of the system using variables that do not depend on the size, known as intensive variables.

Using this approach, we have shown that: $i)$ The density of nodes in an identifier space is an intensive variable that describes a characteristic behavior of a network irrespective of its size. $ii)$ The ratio of perturbation to stabilization β', variable governs the relative amount of wrong pointers and the average lookup path length of the overlay graph irrespective of its size.

In the continuation of this work, we intend to do the following:

- Perform the same experiments with a wider spectrum of numbers to have more statistically-accurate results.
- Use the characteristic behaviors in providing a more adaptive nature to the current DHT algorithms.
- Search for more intensive variables and possible phase transitions.

8 Acknowledgments

We would like to thank Luc Onana Alima and Ali Ghodsi for their efforts in collaborating on the early versions of our implementation of the Chord protocols.

References

[1] *Mozart Consortium*, 2003, http://www.mozart-oz.org.

[2] Luc Onana Alima, Sameh El-Ansary, Per Brand, and Seif Haridi, *DKS(N; k; f): A Family of Low Communication, Scalable and Fault-Tolerant Infrastructures for P2P Applications*, The 3rd International Workshop On Global and Peer-To-Peer Computing on Large Scale Distributed Systems-CCGRID2003 (Tokyo, Japan), May 2003, http://www.ccgrid.org/ccgrid2003.

[3] S. Kirkpatrick and B. Selman, *Critical behaviour in the satisfiability of random boolean expressions*, Science **264** (1994), 1297–1301.

[4] Jinyang Li, Jeremy Stribling, Thomer M. Gil, Robert Morris, and Frans Kaashoek, *Comparing the performance of distributed hash tables under churn*, 2003.

[5] David Liben-Nowell, Hari Balakrishnan, and David Karger, *Analysis of the evolution of peer-to-peer systems*, ACM Conf. on Principles of Distributed Computing (PODC) (Monterey, CA), July 2002.

[6] Ratul Mahajan, Miguel Castro, and Antony Rowstron, *Controlling the Cost of Reliability in Peer-to-Peer Overlayss*, 2nd International Workshop on Peer-to-Peer Systems (IPTPS '03) (Berkeley, CA, USA), February 2003.

[7] D. Malkhi, M. Naor, and D. Ratajczak, *Viceroy: A scalable and dynamic emulation of the butterfly*, InProceedings of the 21st ACM Symposium on Principles of Distributed Computing (PODC '02), August 2002.

[8] Petar Maymounkov and David Mazires, *Kademlia: A Peer-to-peer Information System Based on the XOR Metric*, The 1st Interational Workshop on Peer-to-Peer Systems (IPTPS'02), 2002, http://www.cs.rice.edu/Conferences/IPTPS02/.

[9] M. Mézard, G. Parisi, and R. Zecchina, *Analytic and algorithmic solutions of random satisfiability problems*, Science **297** (2002), 812–815.

[10] D. Mitchell, B. Selman, and H. Levesque, *Hard and easy distributions of sat problems*, AAAI-92. Proceedings Tenth National Conference on Artificial Intelligence (1992), 873, 459–65 (English).

[11] R. Monasson, R. Zecchina, S. Kirkpatrick, B. Selman, and L. Troyansky, *2+p-sat: Relation of typical-case complexity to the nature of the phase transition*, Random Structures and Algorithms **3** (1999), 414.

[12] ______, *Determining computational complexity from characteristic phase transitions*, Nature **400** (1999), 133–137.

[13] Moni Naor and Udi Wieder, *Novel architectures for p2p applications: the continuous-discrete approach*, InProceedings of SPAA 2003, 2003.

[14] Antony Rowstron and Peter Druschel, *Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems*, Lecture Notes in Computer Science **2218** (2001), citeseer.nj.nec.com/rowstron01pastry.html.

[15] I. Stoica, R. Morris, D. Karger, M. Kaashoek, and H. Balakrishnan, *Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications*, ACM SIGCOMM 2001 (San Deigo, CA), August 2001, pp. 149–160.

[16] ______, *Chord: A Scalable Peer-to-Peer Lookup Service For Internet Applications*, Tech. Report TR-819, MIT, January 2002, http://www.pdos.lcs.mit.edu/chord/papers/chord-tn.ps.

[17] Ion Stoica, Robert Morris, David Liben-Nowell, David R. Karger, M. Frans Kaashoek, Frank Dabek, and Hari Balakrishnan, *Chord: A scalable peer-to-peer lookup protocol for internet applications*, IEEE/ACM Transactions on Networking **11** (2003), 17 – 32.

[18] Ben Y. Zhao, John D. Kubiatowicz, and Anthony D. Joseph., *Tapestry: An Infrastructure for Fault-tolerant Wide-area Location and Routing*, U. C. Berkeley Technical Report UCB//CSD-01-1141, April 2000.