Towards Yet Another Peer-to-Peer Simulator

Stephen Naicken Anirban Basu Barnaby Livingston Sethalat Rodhetbhai
Ian Wakeman
Department of Informatics
University of Sussex
{stephenn, a.basu, bjl20, s.rodhetbhai, ianw}@sussex.ac.uk

Abstract

There are a number of P2P overlay simulators developed by various research groups for use by the P2P academic community, however many still opt to use their own custom-built simulator. Having surveyed the area of Peer-to-Peer simulators in previous work, we believe that this is due to the simulators lacking key functionality such as mechanisms to gather statistical data from simulation runs. The use of custom built simulators gives rise to a number of problems that include an increase in the difficulty to reproduce and validate results and comparison of similar simulated systems and their associated results. In this paper, we discuss the current situation with respect to simulation usage in P2P research and our work towards creating a new simulator that will meet the requirements of P2P researchers. It is our hope that this paper will give rise to further discussion and knowledge sharing among those of the P2P and simulation research communities, so that a simulator that meets the needs of the P2P community can be developed.

Keywords: Peer-to-Peer, Simulator Evaluation, Simulator Development

1 Introduction

Peer-to-Peer networks such as Gnutella [1] gained a great deal of notoriety in the late nineties as a mechanism to allow users to share copyrighted material, in particular MP3-encoded music files. While the P2P paradigm was popular among file-sharing users and garnered a great deal of attention from the media, academic researchers began to focus on the benefits, issues and possible application areas of this decentralised, scalable and highly fault-tolerant network structure. Since then a number of problems have been addressed, for example, structured networks have been proposed to allow for greater scalability and faster lookups than unstructured Gnutella-like networks. There is now a great deal of research being conducted on many varied aspects of P2P networks, however, just as with any other scientific research, these solutions must be shown to be valid and other members of the research community must be able to reproduce any results. The methods for achieving this can be analytical, using simulations or by performing experiments with the actual system.

Applying an analytical approach requires mathematically modelling a P2P system, however, such systems tend to be complex, so many simplifying assumptions must be made so that a model can be produced. As a result, any findings may be limited in their applicability, however this approach has been used in this field, for example a theoretical analysis of BitTorrent [2] under concurrent node joins and failures [1].

An alternative to the analytical approach is to run experiments with the actual system. As P2P systems can consist of very large numbers of nodes, performing experiments with the actual system requires significant resources which could be very costly in hardware and administration, and is vulnerable to node failures and version control problems. Introducing malicious nodes into a

[1] http://rfc-gnutella.sourceforge.net/developer/stable/index.html
[2] http://www.bittorrent.com

network for an experiment focused on security issues would also require careful, and possibly limiting, controls. There may also be factors external to the experiment that can not be controlled, yet influence experiment results, such as cross-traffic and changes in the properties of the underlying network layer. Despite these issues, the use of testbeds such as PlanetLab [3] is popular, and it has tools which address some of these issues. There are currently 696 nodes over 336 sites that make up PlanetLab and all these nodes are subject to real-world network conditions. This makes it a particularly useful tool for performing large scale experiments on overlay protocols and for validating results obtained from simulation results.

Simulations of a P2P network do not suffer from the problems given above with analytical methods and experimenting with the system itself. The cost of implementation is less than that of a large-scale experiment as significantly less computing resources are necessary and the model can be less complex than a mathematical one as simulation code can share significant similarities with actual P2P node implementation code. Our survey of Peer-to-Peer simulators [2] raised several concerns. Many of the simulators did not have functionality that we would expect. Almost all had poor documentation that made it difficult to implement P2P well-known overlay algorithms, such as Chord. Of more concern was the fact that some had no mechanism at all to allow for the collection of statistical data from a simulation run. Another issue is that many papers do not give any reference to the simulator that has been used or use a custom built simulator specifically for their work. Lack of clarity in the properties of experiments makes reproducibility of results and analysis and comparison between algorithms problematic. Given these issues with current P2P simulators and the importance of simulation for proving hypotheses and validating and reproducing results, in this paper we discuss our work to date in formulating requirements for a general-purpose P2P simulator and a strategy for implementing this.

The rest of this paper is organised as follows: In section 2, a summary of our survey on P2P simulators [2] is given; section 3 discusses the current state of simulation use in P2P research; in section 4, a set of requirements for P2P simulators is given; and in section 5 a number of simulators are evaluated with respect to these requirements. The paper ends with future work and conclusions in sections 6 and 7 respectively.

2 Summary of PGNET Peer-to-Peer Simulator Survey

In table 2, details of the simulator projects that were surveyed are given. Table 2 provides a comparison of the surveyed simulators. We have omitted some of the simulators from the second table as they are not of importance with respect to this paper (e.g. one is a teaching tool unsuitable for use as simulator for research) and due to space considerations.

Usability and in particular poor documentation is a problem with many of the simulators, but perhaps of more significant concern is the lack of support for statistics. Being unable to gather statistics from a simulator renders it almost useless for research, and those simulators that do support statistics require source code hacking of the simulator.

[3] http://www.planetlab.org

Table 1: Properties of surveyed simulators.

Simulator	**Language**	**Project Activity**	**License**	**Notes & URL**
P2PSim	C++	Active	GPL	`http://pdos.csail.mit.edu/p2psim/`
PeerSim	Java	Active	LGPL	Developed in the BISON project at University of Bologna, Italy. `http://peersim.sourceforge.net/`
Query-Cycle Simulator	Java	Inactive	Apache	Stanford University, USA. `http://p2p.stanford.edu/`
Narses	Java	Inactive	GPL-like	Stanford University, USA. `http://sourceforge.net/projects/narses`
Neurogrid	Java	Inactive	GPL	`http://www.neurogrid.net/`
GPS	Java	Inactive	Open-Source, No License	University of Binghamton, USA. `http://www.cs.binghamton.edu/wyang/gps/`
Overlay Weaver	Java	Active	Apache	`http://overlayweaver.sourceforge.net/`
DHTSim	Java	Active	GPL	Originally developed at University of Pennsylvania, USA. Modified and used in teaching at University of Sussex, UK. `http://www.informatics.sussex.ac.uk/users/ianw/teach/dist-sys/dht-sim-0.3.tgz`
PlanetSim	Java	Active	LGPL	University Rovira i Virgili, Spain. `http://planet.urv.es/planetsim/`

Table 2: Project details of surveyed simulators.

Simulator	**Architecture**	**Usability**	**Scalability (max nodes)**	**Statistics**	**Underlying Network**
P2PSim	Discrete-event for structured P2P networks	Poor documentation	3000 nodes	Limited set of statistics can be collected before coding required	A range, including: end-to-end time graph, G2 graph, GT-ITM, random, and Euclidean
PeerSim	Query-Cycle or Discrete-event for unstructured networks	Only Query-Cycle simulator is documented	10^6 (Query-Cycle)	Components can be implemented to gather statistical data or to simulate nodes joining, departing and failing	Not modelled
Narses	Discrete-event, flow-based for tunable topologies	No documemntation, source code is difficult to understand	600 nodes, depending on the underlying topology, untested by us	Yes, but requires modification of source	A number of underlying topologies, balancing execution speed and accuracy
Overlay Weaver	Distributed Emulation and a number of structured overlay algorithms	API and source code well documented, but some documentation missing	4000 nodes, we obtained only 2700 due to kernel & GLibC thread limits	Not possible to gather statistics	Not modelled
PlanetSim	Discrete-event simulator; dses Common API; distinct seperation between services and overlay	Design and API throughly documented; detailed tutorial	100,000 nodes	No mechanism to gather statistics, but visualiser is available	Limited simulation of underlying network, but BRITE information could be used for more detail
Neurogrid	Discrete-event for unstructured networks, can be modified for use with structured networks.	Extensive documentation on web	300,000 nodes claimed, but replication fails due to thread limits	For pre-determined variables, but code would have to be modified for others	Not modelled

3 Surveying the Use of Simulation in Peer-to-Peer Research

We surveyed a total of 287 papers on peer-to-peer networking subjects to find out which simulator, if any, they use for their experiments. The results are shown in figure 1.

None	146
Unspecified	71
Custom	43
NS-2	8
Chord (SFS)	7
Javasim	2
Peersim	2
Aurora	1
CSIM 19	1
Modelnet	1
Nab	1
Narses	1
Neurogrid	1
P2PSim	1
SOSS	1

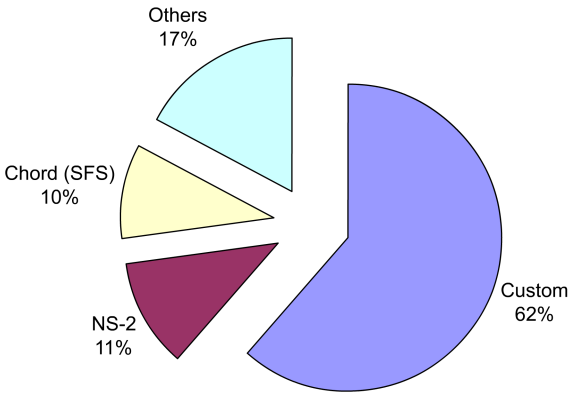

Figure 1: Quantitative survey on the use of simulators

"None" includes papers which involve no simulation. "Unspecified" are papers which talk about simulation and provide results, but do not mention which simulator was used. They might have created one themselves, used an existing simulator but not named or referenced it or they might have used some kind of generic simulator framework. "Custom" is all papers which say that a simulator was created specifically for the paper. Some of these papers go on to describe the simulator at length. "NS-2" includes all papers which used the general, packet-level network simulator NS-2. This program provides extensive low-level simulation of networks and as such is not specifically a peer-to-peer simulator. The papers categorised as "Chord" used a modified version of the SFS based simulator used in Chord [3] [4]. The rest used a named peer-to-peer simulator, some of which are discussed in this paper and some others in our previous paper [2].

More than half of the relevant papers we surveyed mentioned that simulation was carried out as part of their experimentation but failed to specify which simulator was used. This could make it difficult to fully realise the significance of the results and harder to reliably reproduce them.

From these results is it clear that of the 70 papers which stated which simulator they used, 62% of them used a specially created simulator. The pie-chart in figure 1 illustrates this. Some of these simulators might possibly be the same, reused within research groups. However, even taking this into account, the number of custom-made simulators far outnumbers the use of known simulators. This is not an ideal state of affairs, both in terms of duplication of effort and for ease of comparison and replication of results.

4 Peer-to-Peer Simulator Requirements

We shall now present requirements on a general-purpose simulator which arise from our study of the literature and will later be used to evaluate existing simulators for possible development.

Simulation Architecture The two main classes of architecture are discrete event and query-cycle. In discrete event, a scheduler thread synchronises a queue of messages to be transferred between simulated nodes. In query-cycle the simulation loops through each node, carrying out queries for the nodes as required.

Simulation Behaviours The simulator will need to be able to simulate node behaviours such as churn and node failures with good flexibility. For example, it should be possible to have nodes fail at a specified time and then have a user-specified probability of returning after a time. The simulator should be able to simulate both structured and unstructured networks.

Usability The simulator should include a utility that allows the user to create a simulation run. This should be a command line program, but a graphical front-end could also be included. It should be possible to create simulation scripts with a text editor. A utility that converts scripts from other simulators would be useful. The simulator should be able to make use of declarative overlays such as P2 [4] and Mace/Macedon [5]

Documentation The simulator source code should be well commented and should implement a clean, well designed and documented API. Manuals, user guides and other appropriate documentation should be provided. Online support such as mailing lists, newsgroups and websites should be available.

Statistics The user should be able to declare the variables of interest and should be able to collect statistical data about those variables during the run and after the simulation run is complete. The simulator should be able to provide snapshots of its state for analysis. The statistical data output should be easy to manipulate and analyse with tools such as xgraph. No modification of the source code by the user should be necessary to extract statistics. The simulation script file should allow for reproducible experiments.

Underlying Network Modelling The simulator should provide a tunable underlying network model, from totally ignoring the underlying network for maximum scalability, to a low-level model based on output from GT-ITM [5]/Brite [6] for a high degree of real-world accuracy. It should be able to simulate cross-traffic. When simulating an underlying network, the variables associated with the network layer - such as link and packet latencies, link failures and packet loss - need to be dynamic.

System Limitation The simulator should make use of hardware and operating system resources. It should be able to overcome system limitations by distributing the simulation across several computers.

Portability It should be possible to reuse simulation code on PlanetLab with at most minor modification.

Scalability The simulator should be able to run simulations with numbers of nodes in the order of hundreds of thousands. Ideally equalling the claims of Neurogrid (300,000) or PeerSim (10^6).

Debugging The simulator should include an interactive visualiser. This will help the user to understand what is going on in the simulation and to find bugs in the protocol being simulated. It should provide the ability to examine the current state of nodes and links, to pause and resume the simulation, and to take snapshots for detailed analysis.

5 Evaluation of Simulators

Given the survey findings and the list of requirements given above, it is believed that currently none of the surveyed simulators offers a complete solution to our requirements. In this section, three suitable candidate simulators are evaluated with respect to extending one to meet the above requirements, but due to space constraints the reasons for rejecting the other simulators are not given, although it

[4] http://www.cs.binghamton.edu/ wyang/gps/
[5] http://mace.ucsd.edu/, http://macedon.ucsd.edu/

was either due to poor documentation to the extent that the learning curve for the simulator was too steep, rigid simulation architecture that made extension difficult and software or hardware system limitations. The possibility of writing a simulator from scratch is also discussed.

5.1 PlanetSim

PlanetSim [7] is a Peer-to-Peer discrete-event simulator. It is written entirely in Java and is released under the LGPL license as a Sourceforge project. The simulator is under development as part of the Planet project at the University Rovira i Virgili, Spain. It supports both structured and unstructured overlays, but only implementations of two structured overlays, Chord-SIGCOMM and Symphony [8] are distributed with the simulator. In [9], experiment results to demonstrate the scalability of PlanetSim are given. These experiments show that for Chord, 8 seconds is needed to stabalise a 100 node network, 16 minutes for 1000 nodes, and 46 hours for 100000 nodes.

The framework makes use of the Common API [10], so its design is clear and easy to understand. There are three layers to the simulator, Network, Node, Application. The network layer is responsible for the simulation life cycle, the underlying topology and the routing of messages in the simulated network. At the Overlay layer, the overlay algorithm such as Chord is implemented and above this there is the Application layer where P2P services (e.g. DHT primitives) are implemented. By using this layered approach and the Common API, it is possible to implement the different overlays using the same Network layer and the same application using different overlays. This allows for the fair comparison of overlays and P2P services.

With respect to usability, there are a number of tutorial documents for the simulator, the source code is well commented and combined with the fact that it written in Java, the learning curve is not steep. By making use of Java there is the possibility of a performance hit when compared to an implementation of a similar simulator in C, however the authors have claimed to have carefully profiled PlanetSim performance.

Although it is not possible to collect statistics from simulation runs, the simulator supports churn and node failure, has a visualiser that makes use of Pajek/GML, and simulations can be saved to disk for reuse. The authors have suggested further possible improvements to PlanetSim and their is some overlap with our requirements. We are particularly interested in the proposed Network layer wrapper so simulation code can be ported to a real network such as Planetlab.

The underlying network layer can currently be modelled as simple random or circular networks with no consideration of latency costs, cross traffic and bandwidth. It is possible to make use of BRITE network information to produce a more realistic underlying network topology. Ideally, it should be possible to have a tunable underlying network layer, so services and overlay routing algorithms could be tested using varying lower level topologies. A simple topology would be used when emphasis is on the overlay algorithm or P2P service, and the lower network layer could be changed to a more complex one to test the algorithm with more realistic network conditions.

Of all the simulators reviewed, given the current state of PlanetSim and the proposal that the authors have made for future improvements, this simulator should be extended to meet the requirements given above. The PlanetSim project is still active, so there may exist the opportunity to investigate possible collaborations.

5.2 PeerSim

PeerSim [11] was partly developed in the BISON project [6] . It is designed specifically for epidemic protocols (such as OverStat [12, 13], SG-1 [14] and T-Man [15]) with high scalability and support for dynamicity. It can simulate both structured and unstructured overlays. PeerSim avoids many underlying network parameters and simulates an abstraction of the overlay network. PeerSim supports

[6] http://www.cs.unibo.it/bison/

cycle-based and event-based simulation models. The latter model can support concurrency. PeerSim does not support distributed simulation.

The component libraries were mainly designed to suit graph-like overlay protocols. Thus, some modification of existing components or coding of additional components have to be taken into account for implementation of other protocols. The design of PeerSim supports modular programming based on objects. However, in several cases, the pre-defined interfaces or modules may contain some declarations (properties and methods) that are neither relevant nor suitable for the implementation of a certain protocol. Implementation of an overlay algorithm requires the algorithm to be altered to comply with the simulator's interface. The cycle-based model provides less realistic but more efficient simulation than the event-based model.

PeerSim is configured using a plain text file. Scheduling and parameter values for each of the components can be adjusted, but there are certain limitations (such as simple scheduling for the scheduler component) requiring code modification. PeerSim provides neither a graphical user interface nor any debugging facilities. Although it offers components for common statistics, additional coding for user-defined data collection is still necessary.

The PeerSim developer website provides tutorials and examples for the cycle-based model only although there is good API documentation. Due to the lack of comprehensive documentation of the event-based model, it is not straightforward to utilise this model. All these shortcomings have made us decide not to extend PeerSim.

5.3 Overlay Weaver

Overlay Weaver [16] is intended to be a toolkit for easy development and testing of P2P protocols. It provides functionality for simulating structured overlays only and does not provide any simulation of the underlying network. It is packaged with implementations of Chord [3] [4], Kademlia [17], Pastry [18], Tapestry [19] and Koorde [20]. RPC can either be emulated using real TCP/UDP so the protocol can be tested on a real network, or using discrete event message passing within the JVM. Distributed simulation is possible, however it is barely documented.

Documentation in general is quite sparse and does not cover many of the simulator's functions. However the API is clean and well designed so the source is quite readable.

The documentation says that Overlay Weaver has been seen to scale to 4,000 nodes. In experiments it was found that a maximum of 3,300 nodes could be simulated before hitting the maximum number of threads limit in Linux. The maximum number of threads in the kernel was increased, as was the user processes limit. The latter allowed a small increase in the number of nodes possible. The limit being hit now is most likely set in glibc and will require a recompile to increase.

Overlay Weaver has a lot of shortcomings when it comes to simulation. Statistics gathering needs a lot of work, documentation needs to be written and the lack of scalability might need a fundamental redesign. Hence we have decided not to extend it.

5.4 New Simulator from a New Codebase

Following our PGNet survey, the idea of writing a simulator from a new codebase did occur to us. However, we have chosen not to go down this route. The main problem with a new codebase is that of re-inventing the wheel when there are feasible options of re-using existing simulators. Also, writing a new simulator raises the question of its validation and hence its acceptability in the research community. Extending an existing simulator has the added advantage of larger community effort (as long as the simulator being extended is open-source) in developing and bug-fixing the core code of the simulator on top of which we build necessary tools to meet our simulation requirements.

6 Future Work

Given that a year has passed since the last release of PlanetSim (version 3.0 Release Candidate - July 2005), the PlanetSim source code from the project's Sourceforge hosted CVS repository will be forked and a new project created from this codebase. In keeping with the PlanetSim licencing agreement, the GNU Lesser General Public License (LGPL) will be used. This project will be hosted using Google code hosting so that all interested developers can access the source code and contribute to it using Subversion. A roadmap for the development of the simulator must also be drawn up. Simulator requirements that we have given in this paper, must be prioritised and we will actively encourage members of the P2P and Network Simulation research communities to contribute to achieving these objectives.

7 Conclusion

All of the P2P simulators surveyed have functionality missing which we believe is of importance. While poor documentation is a hindrance, it is a problem that can be overcome, but it is entirely unacceptable that many of the simulators have no mechanism to allow a user to gather statistics of a simulation run. We believe that the poor state of existing P2P simulators is the reason that much published research makes use of custom built simulators. Surprisingly, 10% of papers making use of simulation from our survey used NS-2, a tool that is often inappropriate for simulating P2P networks as it lacks scalability due to its detailed modelling of the lower network layers that are often of little interest to P2P researchers. With custom simulators being so popular, it complicates the task of validating research and reproducing results, as these simulators are often not released publicly. Even if they are available comparing similar work, such as overlay algorithms, can be complicated.

Given the current state of simulation use in P2P, we believe that there is a need for a P2P simulator that meets the requirements of P2P researchers. We believe that this is best done by extending the PlanetSim simulator to meet our requirements, requirements that we believe to be similar to those of other P2P researchers. The next priority is to implement a roadmap for the project and to seek feedback and contributions from both the P2P and Simulation research communities regarding the challenges that lie ahead.

References

[1] D. Qiu and R. Srikant, "Modeling and Performance Analysis of BitTorrent-Like Peer-to-Peer Networks," in *SIGCOMM '04:*. New York, NY, USA: ACM Press, 2004, pp. 367–378.

[2] S. Naicken, A. Basu, B. Livingston, and S. Rodhetbhai, "A Survey of Peer-to-Peer Network Simulators," *Proceedings of The Seventh Annual Postgraduate Symposium, Liverpool, UK*, 2006.

[3] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications," in *SIGCOMM '01*. New York, NY, USA: ACM Press, 2001, pp. 149–160.

[4] I. Stoica, R. Morris, D. Liben-Nowell, D. Karger, M. Kaashoek, F. Dabek, and H. Balakrishnan, "Chord: A Scalable Peer-to-Peer Lookup Protocol for Internet Applications," *IEEE/ACM Transactions on Networking*, vol. 11, no. 1, pp. 17–32, 2003.

[5] "GT-ITM: Modeling Topology of Large Internetworks." [Online]. Available: http://www-static.cc.gatech.edu/fac/Ellen.Zegura/graphs.html

[6] "BRITE - Boston University Representative Internet Topology Generator." [Online]. Available: http://www.cs.bu.edu/brite/

[7] U. R. i Virgili, "PlanetSim: An Overlay Network Simulation Framework," 2006, accessed 01-May-2006. [Online]. Available: http://planet.urv.es/planetsim/

[8] G. Manku, M. Bawa, and P. Raghavan, "Symphony: Distributed Hashing in a Small World," in *4th USENIX Symposium on Internet Technologies and Systems*, 2003, pp. 127–140.

[9] P. García, C. Pairot, R. Mondéjar, J. Pujol, H. Tejedor, and R. Rallo, "A New Overlay Network Simulation Framework," *Lecture Notes in Computer Science (LNCS), Software Engineering and Middleware (SEM)*, vol. 3437, pp. 123–137, 2005.

[10] F. Dabek, B. Zhao, P. Druschel, J. Kubiatowicz, and I. Stoica, "Towards a Common API for Structured Peer-to-Peer Overlays," *Proceedings of IPTPS*, vol. 58, 2003.

[11] "PeerSim P2P Simulator," 2005, accessed 01-May-2006. [Online]. Available: http://peersim.sourceforge.net/

[12] M. Jelasity and A. Montresor, "Epidemic-Style Proactive Aggregation in Large Overlay Networks," in *Proceedings of the 24th International Conference on Distributed Computing Systems (ICDCS'04)*. Tokyo, Japan: IEEE Computer Society, Mar. 2004, pp. 102–109.

[13] A. Montresor, M. Jelasity, and O. Babaoglu, "Robust Aggregation Protocols for Large-Scale Overlay Networks," in *Proceedings of the 2004 International Conference on Dependable Systems and Networks (DSN'04)*. Florence, Italy: IEEE Computer Society, June 2004, pp. 19–28.

[14] A. Montresor, "A Robust Protocol for Building Superpeer Overlay Topologies," in *Proceedings of the 4th International Conference on Peer-to-Peer Computing (P2P 2004)*. Zurich, Switzerland: IEEE, Aug. 2004, pp. 202–209.

[15] M. Jelasity and O. Babaoglu, "T-Man: Gossip-based Overlay Topology Management," *Engineering Self-Organising Applications*, 2005.

[16] K. Shudo, "Overlay Weaver," 2006, accessed 01-May-2006. [Online]. Available: http://overlayweaver.sourceforge.net/

[17] P. Maymounkov and D. Mazières, "Kademlia: A peer-to-peer information system based on the xor metric." in *IPTPS*, 2002, pp. 53–65.

[18] A. I. T. Rowstron and P. Druschel, "Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems." in *Middleware*, 2001, pp. 329–350.

[19] B. Zhao, J. Kubiatowicz, and A. Joseph, "Tapestry: An Infrastructure for Fault-Tolerant Wide-Area Location and Routing," *Computer*, 2001.

[20] M. Kaashoek and D. Karger, "Koorde: A Simple Degree-Optimal Distributed Hash Table," *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS '03)*, 2003.