PEER-TO-PEER NETWORK SIMULATION

Nyik San Ting, Ralph Deters
Departmentr of Computer Science, University of Sasjatchewan, 57 Campus Drive, Sasktaoon, 5A9S7 Canada

Keywords: Peer-to-Peer Networks, Simulation

Abstract: Peer-to-Peer (p2p) networks are the latest addition to the already large distributed systems family. With a strong emphasis on self-organization, decentralization and autonomy of the participating nodes, p2p-networks tend to be more scalable, robust and adaptive than other forms of distributed systems. The much-publicized success of p2p-networks for file-sharing and cycle-sharing has resulted in an increased awareness and interest into the p2p protocols and applications. However, p2p-networks are difficult to study due to their size and the complex interdependencies between users, application, protocol and network. This paper has two aims. First, to provide a review of existing p2p-network simulators and to make a case for our own simulator named 3LS (3-Level-Simulator). Second, it presents our current view that there is a need for more realistic/complex models in p2p-network simulation since ignoring the underlying network, topology and/or the behaviour of applications can result in misleading simulation results.

1 INTRODUCTION

The field of P2P networks is still undergoing major changes with new applications and protocols emerging on a nearly monthly basis. However, due to the difficulties in evaluating them prior to their large-scale deployments, they are often short-lived – disappearing as fast as they emerge – normally due to bad performance. What works well in a controlled lab environment, using a small number of nodes, high bandwidth, low latency and highly cooperative users often fails in real world deployments due to lack of bandwidth, heterogeneity and uncooperative users.
Testing a system's performance prior to its deployment is a fairly common element in the software development of applications. Pawlikowski (Pawlikowski, 2002) identifies two main possible experimentation streams: experimentation with the actual system and experimentation with a model (physical or abstract) of the system.

P2P networks tend to be large, heterogeneous systems with complex interactions between the physical machines, underlying network, application and users. Hence, testing of a "running" p2p-network or protocol in a realistic environment is often not feasible. However, it is possible to use a simulation of a p2p-network to evaluate the applications and protocols in controlled environment. Since a simulation requires the creation/definition of a model that serves as an abstraction of the real p2p-network, the issues of model-scope and model-detail arise. What should be included and what can be ignored? Can the physical network be ignored? Is the topology of the p2p-network important? And last but not least, what workloads should be chosen? Especially with the emergences of more complex p2p protocols and applications that emphasize self-organization due to network, loads or past experiences, this issue is becoming more important. Consequently, a p2p simulation model needs to reflect the dependencies between the users, application, p2p-protocol, p2p-network topology and physical network.

The remaining part of this paper is structured as follows. Section 2 reviews existing p2p-network simulators. Based on the shortcomings of existing simulators section 3 presents a novel multi-level simulator called 3LS (3-Level-Simulator). This simulator is evaluated in section 4 and the paper concludes with a summary in section 5.

2 P2P NETWORK SIMULATORS

As mentioned earlier, p2p-network simulators are one option in studying the behaviour of P2P systems. To date only a few p2p simulators have been implemented to aid the research on application and protocols development.

San Ting N. and Deters R. (2004).
PEER-TO-PEER NETWORK SIMULATION.
In *Proceedings of the Sixth International Conference on Enterprise Information Systems*, pages 84-91
DOI: 10.5220/0002624800840091
Copyright © SciTePress

2.1 Evaluation Criteria

There are two necessary conditions for obtaining credible results from a simulator: using a valid simulation program and executing a valid simulation experiment. “A simulation program is valid, if it is a verified computer program of a valid simulation model” (Pawlikowski, 2002). A validated simulation model is a model that is a satisfying accurate approximation of the system under study.

Table 1 contains the criteria considered important in evaluating a P2P simulator.

Table 1: Criteria

Criteria	Definitions
Usability	How easy are the use of the simulator, preparation of input data and the extraction of simulator data?
Extensibility	How easy can the simulator be extended to handle modified/new models e.g. new p2p protocol?
Configurability	How easy is the customization of the simulation?
Interoperability	Can the simulator be used to interoperate with other application?

2.2 Serapis

Serapis (Sandberg, Serapis, 2001) is possibly one of the earliest p2p-network simulators. It was designed to allow the evaluation of different *caching* algorithms for the FreeNet protocol. Serapis has been extended over time to simulate the Gnutella protocol, however, the work is halted and the extension is not yet completed (the latest update was made in November 2001). Serapis focused on the “static network designs with different connectivity patterns and routing algorithms” (Joseph, 2001). It has been shown that the results of simulations on FreeNet obtained using Serapis were inaccurate and that the simulator fails to simulate the actual stresses and strains of a live deployment in an accurate manner (Joseph, FreeNet, 2001).

2.3 NeuroGrid Simulator

NeuroGrid (Joseph, 2001) (Neurogrid, 2003) is a p2p-network simulator designed to study search operations for FreeNet, Gnutella and the NeuroGrid protocols. The NeuroGrid simulator is a single-threaded discrete event simulator that can be customized using configuration/properties files. Among the parameters users can control are, the type of protocol to simulate, the number of searches to simulate and the preferred user interface. The simulation data (e.g. number of messages parsed and the states of the simulation) can be saved into files for later analysis. NeuroGrid assumes that the distance between nodes is constant and ignores latency, congestion and bandwidth issues. After a search message is sent out, the nodes that received the messages take turns in forwarding the messages (due to the single-threaded design of the simulator). After the current search message has been served is it possible to launch another new message (sequential execution of search requests).

NeuroGrid is still very much a work in progress and efforts are made to improve the level of detail in the network models. In its most recent form, the simulator enables the users to specify the number of nodes to simulate (this is also the number of nodes to add to the current simulation after a number of searches is done), the initial number of connections for each node, the number of searches to be generated, and the initial network topology (only ring or at random networks).

Since NeuroGrid is designed to simulate the searching algorithms of different protocols, it is necessary to let the user specify the number of keywords used for the simulation, the number of the documents used for the simulation, the number of keywords per document and the number of documents stored on each node (document and keyword assignments are all randomized). The latest release of NeuroGrid (version 0.1.4) has included the simulation of resource-limited nodes and introduced the concept of the dishonest node.

The simulator can be customized to needs of a user-defined p2p-application based on the above-mentioned protocols by extending the classes provided.

2.4 FreeNet Simulator

The FreeNet simulator (Pfeifer, 2002) was designed to analyze different caching algorithms for the FreeNet protocol. It uses a two-steps mechanism to support the event handling allowing multiple messages to be sent at a time. In the first step, the simulator will move the messages from a temporary storage space to a queue of the node that will process them in the next iteration. In the second step, the simulator will process the messages queued at each node (previous iteration) and put the newly generated messages in the temporary storage space.

With this design, all the nodes act synchronously without mixing the newly arrived messages and the old messages. The user can modify the factors for the simulation by manipulating an interface class that is implemented by the other classes. This design requires that the simulator source code be recompiled each time after the parameters of the simulation have been changed (!).

The user can adjust the maximum number of nodes to simulate, the TTL of the messages, the type of nodes to be simulated (the caching algorithm), the probability of a request event for a file at each node and the probability of faulty information insertion by the node. Unfortunately, the handshaking between nodes is not implemented in the current version of the simulator.

In order to initialise a network, the simulator is started with three connected nodes and new nodes are added iterative until the user-defined number of nodes to simulate is reached. The simulator allows nodes only to be added assuming a static network. After the network is initialised with the number of nodes desired, the desired number of files will be inserted into the network. After this the simulator will be started with initiating the request events.

Output of the simulator such as number of attempted and successful actions for file-insertions and searches are printed onto the command-line.

2.5 FreePastry

FreePastry (FreePastry) is an open-source implementation of the Pastry protocol in Java that can be used to emulate a Pastry network. The latest release of FreePastry (January 28, 2003) includes the implementation of the PAST (Rowstron, 2001) archival storage system that is based on the Pastry protocol and an implementation of the Scribe (Castro, 2002) group communication infrastructure.

The settings of the FreePastry parameters, such as the number of nodes to simulate and the number of events to generate, is done by providing the values in the command line upon starting the simulator. The results are displayed on the command prompt screen as the messages are being processed. Since the Pastry routing uses proximity metrics, it is necessity to represent the proximity in the simulation. Random, Euclidian and sphere are currently supported in FreePastry. In the Euclidean network topology the nodes are randomly placed in a Euclidean plane and the proximity is based on the Euclidean distance in the plane. Whereas, in the Sphere Network topology, the nodes are randomly placed on a sphere, and the proximity is based on the Euclidean distance on the sphere. However, the network delay for the message passing is not simulated, as the simulator is not designed to simulate time.

2.6 Summary

Current p2p-network simulators are limited in their use, difficult to customize and generally tend to ignore the physical network and the user behaviour. The simulators do not support the customisation of the initial network state (connections between the simulated computers and the network delay) and are limited in the level of detail and the scalability of the supported models. Furthermore, the simulators are mostly focusing on the caching algorithms and ignoring the fact that other activities can also impact the efficiency of the system.

While the NeuroGrid simulator is providing good network visualization it does not simulate the user events, network latency and the processor delay of the nodes. Hence the simulation does not reflect the real world situation, especially with the serial searches functionality. Another tricky issue in using of FreePastry and NeuroGrid is the serial fashion in which they execute search events. Due to the absence of a GUI in the FreeNet and FreePastry simulators the modification of parameters is cumbersome. Some of the settings are made through the command line and some have to be encoded in the program.

The FreeNet simulator supports synchronous actions of nodes but fails in providing support for modeling the network latency and the user's behavior. In addition the concept of recompilation after changing is rather crude and limits the use significantly.

Adapting the simulator to new or modified protocols is a question of great practical importance. NeuroGrid and FreeNet do not simulate the network overlay, and hence it is hard to extend the simulation to handle new protocols that need the network proximity information, e.g. the Pastry protocol.
On the other hand, though FreePastry is focused on the Pastry protocol, the simulator is more decoupled and the code can be reused and extended to implement a different protocol e.g. Gnutella. However, the serialized event handling and the absence of simulation time make it difficult to simulate the network delay and processor delay.

Table 2: Evaluating existing P2P Simulators

	Neuro-Grid	FreeNet	FreePastry
Event-processing	Serial	Parallel	Serial
Usability	Very easy	Medium	Hard
Extensibility	Medium	High	Medium
Configurability (Easiness)	Mid-High (High)	Medium (Medium)	Low (Mid-Low)
Interoperability	Medium	Medium	High
Level of Detail	Medium	High	High
Build-ability	Medium	High	Very High
Simulating User behavior	No	No	No
Simulating Computer Hardware	No	No	No
Simulating network overlay	No	No	Yes
Simulating time (Network delay)	No (No)	Yes (No)	No (No)

It is therefore our conclusion that none of the existing simulators are suited for realistic simulations of different p2p-networks and that the development of a new simulator is therefore justified.

3 TOWARDS AN OPEN MULTILEVEL P2P SIMULATOR

Researchers, who wanted to simulate a p2p system, tend to avoid the development of a complex simulator and focus on some selected areas (such as caching schemes). While some may choose to start an implementation from scratch, an increasing number of researchers build their simulators on top of existing tools, e.g. the agent platform JADE (Bellifemine, 1999), to speed-up the development. The general problem of having only special-purpose simulators is that the results obtained with one simulator are difficult to validate and often impossible to achieve with another simulator due to the many hard-coded assumptions of every simulator.

This section presents an architecture and implementation of an open p2p simulator, called 3LS (3-Level-Simulator), designed to overcome the problems of existing simulators namely, extensibility, usability and level of detail. Since the development of a simulator is a complex and time-consuming activity we haven't completed all parts of the simulator and in its current version it supports only Gnutella-style protocols. The simulator is currently used to help in the development of Comtella (Vassileva, 2002) a p2p file-sharing applications for sharing research papers.

3.1 Design Goals

The criteria used to evaluate the simulators in section two were used as a guideline for the design of a more generic and open simulator that will allow users to define models for the physical network, the physical machines, p2p-network topology, p2p protocol, p2p application and user behavior.

3.2 Architecture

Figure 1 shows a high-level view of the 3LS simulator. 3LS is a time-stepped simulator that uses a central step-clock. In 3LS the models for network, p2p protocol and user model are clearly separated. This separation allows the simulation of various network topologies, for different protocol, applications and user models. To achieve this separation three levels have been defined:

- Network level (bottom),
- Protocol level (middle) and
- User level (top).

Upon starting the simulator it is possible to either create the models for the three described levels (fig. 2) or to choose among a library the ones most suited for the simulation run. As the simulation is running, the events are displayed on the command prompt screen. After the simulation has been completed, all simulation data is saved into a file for future analysis.

As Pawlikowski points out, either general-purpose languages (such as FORTRAN, Pascal, C and Java) or simulation languages (such as GPSS, SIMAN or SLAM II) can be used for the "translation of the model into a computer program" (Pawlikowski, 2002). Though simulation languages provide most of the features needed in programming a simulation model and the details of the simulation models can be easily changed, a general-purpose language was selected to provide "greater programming flexibility".

Since Java is the preferred language of many p2p programmers it was chosen as the host-language for the 3LS simulator.

Visualization of the network is done with the aid of the tool AiSee (AbsInt). AiSee was selected for its, ease of use, simple installation, availability (runs under various OS), functionality and performance in rendering. When screenshots of the p2p-network are to be visualized, files containing the information of the graph are created by 3LS using the Graph Description Language (GDL).

Once the file is created a user can use AiSee to render an image of the graph (see figure 3).

3.3 Network Level

Using the GUI (fig.3) or predefined scenarios, the network level creates a two-dimensional matrix storing the distance values between the nodes. The network level is responsible for modeling the user-defined aspects of a physical network e.g. varying network load due to increased P2P communication. The network-level also creates a user-defined number, of computer nodes with a user defined-number of worker threads associated that take care

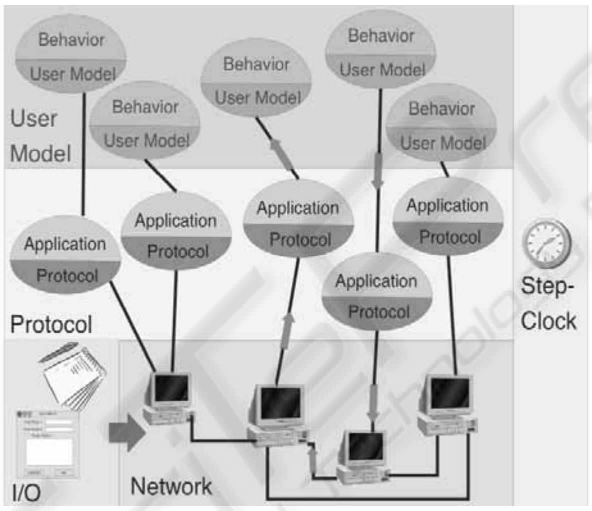
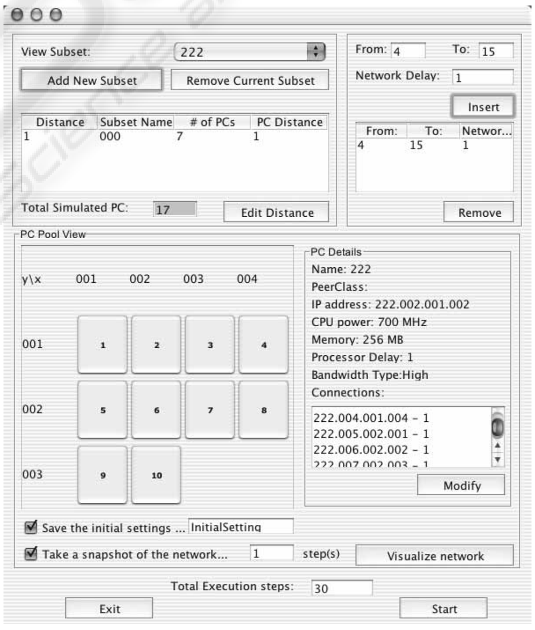


Figure 1: Architecture of the 3LS



View Subset: 222

Distance	Subset Name	# of PCs	PC Distance
1	000	7	1

Total Simulated PC: 17

From: 4 To: 15

Network Delay: 1

From:	To:	Networ...
4	15	1

PC Pool View

y\x	001	002	003	004
001	1	2	3	4
002	5	6	7	8
003	9	10		

PC Details
Name: 222
PeerClass:
IP address: 222.002.001.002
CPU power: 700 MHz
Memory: 256 MB
Processor Delay: 1
Bandwidth Type:High
Connections:
222.004.001.004 – 1
222.005.002.001 – 1
222.006.002.002 – 1
222.007.002.003 – 1

☑ Save the initial settings ... InitialSetting

☑ Take a snapshot of the network... 1 step(s)

Total Execution steps: 30

Figure 2: Screenshot of 3LS – Network-Model.

of the messages passing and the processing of the messages at application node. By varying the number and the priority of the worker threads the processor load and simulation speed of the tool can be adjusted. Each node in the network level represents a computer with user-defined hardware specification that is used to simulate in a more manner accurate individual machines. Interactions between the network level and the protocol level are made through referencing the application nodes in the protocol-level. Each node used four queues for handling message objects:

- Outbox,
- Inbox-For-Network-Delay,
- Inbox-For-Processor-Delay and
- Inbox.

To illustrate the way the nodes between different layers work, a simple example of sending messages between nodes of the protocol layer will be used. Sending a message from one application node (X) to another one (Y) is done as follows:

i. A message object is created, time-stamped and placed by the application node (protocol layer) in the outbox of the computer node X (network layer). A worker thread responsible for checking the outboxes will detect the message and move it from the outbox of node X into the inbox-for-network-delay of node Y.

ii. After checking the outboxes of all nodes the global simulation clock is incremented and the worker threads start checking the message objects in the inbox-for-network-delay using the 2-D distance matrix and the congestion network delay data. The goal is to simulate the delay/latency of the network by postponing the delivery of message. If the assumed network delay has been fulfilled, the message will be stored in the inbox-for-processor-delay that serves as a means to simulate the time a node needs for processing the message.

iii. After the simulation clock is incremented, the worker threads will look at the processor delay of the node Y and check whether the message object in the inbox-for-processor-delay can be moved into the inbox for the application node. In case the processor delay has not been reached, the message object remains in the inbox-for-processor-delay.

iv. Once the delay has passed the message object will be sent to the appropriate application node. After the application node processed the received message object, it will perform user-defined responses. If the response results in the creation of new messages they are stored in the outbox and the process starts again with the step i.

The simulator follows a 2-steps mechanism: for each unit of user-time, it takes 2 step-times in the simulation. Figure 5 shows the actions performed at step t. With this design, the network delay and processor delay can be simulated, and the message arriving is simulated in a more realistic manner. In addition this design enables several tasks (or events) being carried out at any time. At any moment in time a user can request that the simulator generates a visualization of the current network (fig. 3). The information encoded in the network snapshot includes the events (messages traversed) during the step, the status of the computer nodes (variables such as the memory) and the network connections at the end of the step.

3.4 Protocol Level

A special class (peer interface class) is used to provide an interface for the worker threads in the network level and to enable the sending of messages from a computer node to an application node. Any protocol implementation has to implement this interface class. To create application nodes, the user needs to specify the IP address and port number of the peers created. Upon being created, the application node/peer can use the registration class provided by the network level to register itself to a port of the computer node using the IP address provided. The implementation of the message object is important in this simulator. The message object contains the time-stamp, reference to the message content object, the origin’s IP address and port number, and the destination IP address and port number.

3.5 User Level

A user model contains the method signatures for the decision-making needed from the protocol level and is linked to a specific peer instance adding tasks into the task-scheduler of its peer node. The user model serves as a load-generator and state-based controller of the peer nodes allowing for a more accurate modeling of the behavior of peers.

4 EVALUATION

In its current implementation, the simulator consists of 29 classes with approximately 3599 lines of code. There are 10 classes (899 lines of code) for the platform level, 12 classes (508 lines of code) for the Gnutella protocol and user task scheduling, and 7 classes (2159 lines of code) for GUI interfaces including the main method class. Using the Gnutella 0.4 protocol a series of performance and accuracy tests were conducted to evaluate 3LS. We used AMD Athlon Processor 800Mhz with 524 MB of RAM running Microsoft Windows 2000 and a Sun SunFire3800 with 4 UltraSparc III CPUs running at 750 MHz and 8 GB of RAM running Solaris 8.2. For each simulation with n nodes, there is a total of (n-1)+(n-1)*(n-2) number of events (message sent from a peer to the other).

The test revealed that the simulation of medium-sized p2p-networks consumes already significant CPU resources and memory. The memory consumption increases with the number of events and is due to the saving of the messages (texts showing the events occurred in the simulation) in a hashtable until the simulation is finished upon which data is saved into a file. To be able to scale up to higher numbers of simulated peers it is necessary to distribute the simulation over multiple processes/host.

5 Future Work

Future work focuses on collecting data for the various layers e.g. human desktop usage and network traffic. We are currently testing the simulator by comparing its results for a Gnutella 0.4 network (Clip2) with the "real data" obtained from running Gnutella 0.4 clients in a controlled network. Using Comtella (Vassileva, 2002) clients we are able to adjust the various parameters of the simulation and verify the simulation results. Early results in a small network (less than 20 nodes) indicate that the simulator works as expected but more testing is needed.

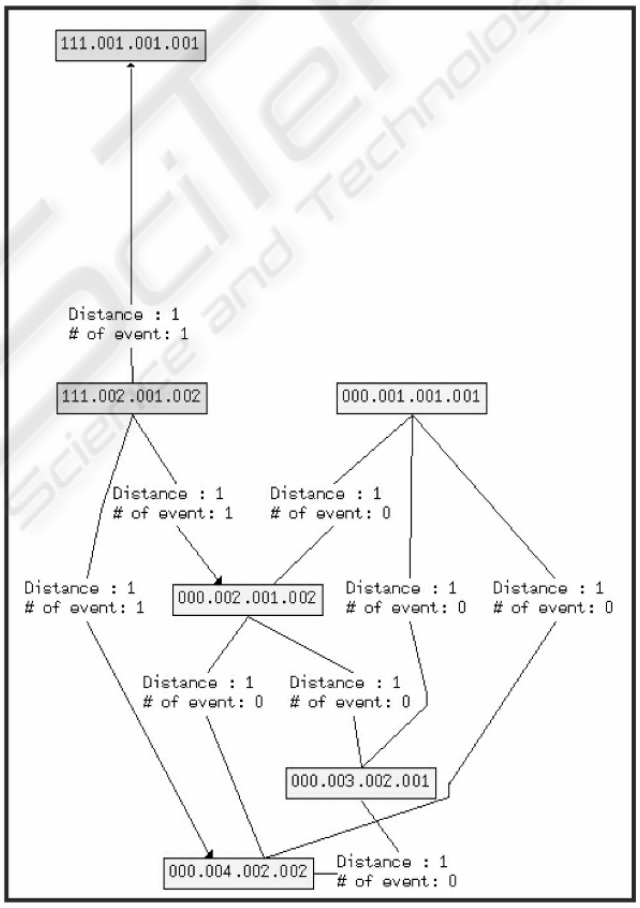


Figure 4: Example of network view using AiSee.

6 CONCLUSION

This paper provides an overview of existing P2P protocols, examines the simulators and proposes a generic P2P simulation model. The simulator enables the simulation of P2P networks with different network topology, user models, and applications. We hope that by starting the development of a open java-based p2p-network simulator the community of p2p developers and researcher will be able to develop models and loads that enable the evaluation of current and future protocols.

7 CODE

The complete code of the 3LS simulator is available upon request by sending an email to one of the authors. 3LS requires a Java 1.3.1 or higher version of the JDK.

REFERENCES

AbsInt. AiSee homepage. http://www.aisee.com/

Bellifemine, F., Poggi, A., Rimassa, G. JADE–A FIPA-compliant agent framework In Proceedings of PAAM'99,London, April 1999, 97-108.

Castro, M., Druschel, P. Kermarrec A-M., Rowstron A. SCRIBE: A large-scale and decentralised application-level multicast infrastructure. IEEE Journal on Selected Areas in Communications (JSAC) (Special issue on Network Support for Multicast Communications), to appear, 2002.

Clip2. The Gnutella Protocol Specification v0.4. http://rfcgnutella.sourceforge.net/Development/GnutellaProtocol0_4-rev1_2.pdf

FreeNet. http://freenet.sourceforge.net.

FreePastry. The FreePastry homepage. http://www.cs.rice.edu/CS/Systems/Pastry/FreePastry/

Joseph, S.R.H. Adaptive Routing in Distributed Decentralized Systems: NeuroGrid, Gnutella, and Freenet. Proceedings of workshop on Infrastructure for Agents, MAS, and Scalable MAS, at Autonomous Agents, Montreal, Canada, 2001.

Joseph, S.R,H. Project:NeuroGrid -P2P Bookmark Organiser:Mailing Lists. NeuroGrid Simulation Mailing Archive, Jan 2003 http://sourceforge.net/mailarchive/forum.php?thread_id=1593250&forum_id=8271

NeuroGrid. The NeuroGrid homepage. http://www.neurogrid.net/

Pawlikowski, K. Simulation Modeling and Analysis with an emphasis on applications in performance evaluation of telecommunication networks. Course 410, University of Canterbury, August 2002. http://www.cosc.canterbury.ac.nz/teaching/handouts/cosc410/02.410.notes1.pdf

Pfeifer, J. Freenet Caching Algorithms Under High Load. http://www.cs.usask.ca/classes/498/t1/898/W7/P2/freenet.pdf, 2002.

Rowstron, A., Druschel, P. PAST: A large-scale, persistent peer-to-peer storage utility. HotOS VIII, Schoss Elmau, Germany, May 2001.

Sandberg, O. The FreeNet-dev mailing list, March 2001 http://www.ultraviolet.org/mail-archives/freenet-chat.2001/0354.html

Serapis. The Serapis homepage, cvs. Sourceforge.net. http://cvs.sourceforge.net/cgi-bin/viewcvs.cgi/freenet/Serapis/

Vassileva, J. Motivating participation in Peer to Peer Communities. Proceeding of Workshop on Emergent Societies in the Agent World, ESAW'02, Madrid, 16-17 September, 2002.