Author(s): Kotilainen, Niko; Vapa, Mikko; Keltanen, Teemu; Auvinen, Annemari; Vuori, Jarkko

Title: P2PRealm - Peer-to-Peer Network Simulator

Year: 2006

Version:

Please cite the original version:

Kotilainen, N., Vapa, M., Keltanen, T., Auvinen, A., & Vuori, J. (2006). P2PRealm - Peer-to-Peer Network Simulator. In Proceedings of 11th International Workshop on Computer-Aided Modeling, Analysis and Design of Communication Links and Networks (pp. 93-99). https://doi.org/10.1109/CAMAD.2006.1649724

P2PRealm – Peer-to-Peer Network Simulator

Niko Kotilainen, Mikko Vapa[+], Teemu Keltanen, Annemari Auvinen[+] and Jarkko Vuori
Department of Mathematical Information Technology, University of Jyväskylä
P.O.Box 35 (Agora), 40014 University of Jyväskylä, Finland
firstname.lastname@jyu.fi

***Abstract*—Peer-to-Peer Realm (P2PRealm) is an efficient peer-to-peer network simulator for studying algorithms based on neural networks. In contrast to many simulators, which emphasize on detailed network simulation, the speed of simulation in P2PRealm is essential, because neural networks require a time consuming training phase. Efficiency has been obtained by optimizing training loops inside the simulator, using Java Native Interface (JNI) as well as distributing the simulator to hundreds of workstations using the P2PDisCo platform. In this paper we describe the architecture of P2PRealm and its input/output interfaces. Also, we present the mechanisms used for internally optimizing the implementation and the configuration used for distribution. Finally, we present the use of P2PRealm with the P2PStudio network visualization tool.**

Keywords - peer-to-peer; P2PRealm; network simulation; research infrastructure; neural networks, optimization methods;

I. INTRODUCTION

Peer-to-Peer (P2P) algorithms have been studied at least using three different approaches. These are crawlers, emulators and simulators. Crawler is an implementation of a peer node specially designed for P2P networks research. A crawler can collect data passing through it to get a local view of the P2P network. By deploying multiple crawlers, a bigger part of the peer-to-peer network can be monitored. However, this approach is not able to gather the global view of the network, because the behavior of nodes which are not connected to crawlers is unknown. This problem is solved by emulators, which contain the implementation of a peer node and are used to build a complete P2P network. Multiple emulators can be deployed inside one workstation usually providing quite large P2P networks with only a few workstations.

Even though emulators can be used to get the global view, they are restricted to slow execution, because messages need to be passed between emulator processes through network protocols such as TCP. The third option, simulator, contains an abstracted implementation of peer nodes equivalent to emulators, but uses local data structures for message passing. The use of local data structures significantly increases the speed of execution and therefore is well-suited approach for computationally intensive algorithm studies. The downside of the approach is the inaccuracy of results compared to real-world P2P networks and the difficulty of modeling user behavior. This knowledge can only be obtained using crawlers or by monitoring network traffic inside routers.

Developing peer-to-peer resource discovery and topology management algorithms based on neural networks are computationally intensive tasks. For example, it takes a week for one low-cost workstation to train a good neural network based resource discovery algorithm for a rather small P2P network [23]. Therefore, in computationally intensive algorithmic research the most important factor to consider for a simulator is speed.

This paper describes an end product of a process where emulators were first used for studying P2P algorithms and later re-implemented as an efficient simulator to decrease the time used for execution. Latest improvement of the simulator is the distributed execution on a Peer-to-Peer Distributed Computing platform (P2PDisCo) [11] allowing us to parameter sweep different features of neural network based P2P algorithms.

The paper is structured as follows. Section 2 compares existing P2P simulators with our work and states their differences. Section 3 introduces P2PRealm network simulator and section 4 describes its input and output interfaces. Section 5 describes the main use case of P2PRealm: the training of evolutionary neural networks for P2P resource discovery. Section 6 describes the internal modifications used for optimizing the code and the combination of P2PRealm with P2PDisCo platform. Section 7 illustrates the use of P2PStudio for visualizing the output of P2PRealm and section 8 concludes the paper.

II. RELATED WORK

There are various network simulators available for studying P2P networks. However, many of these simulators are not primarily designed for speed and none of them contains functionalities for neural networks. Because the speed is the most important factor in our simulation environment, it is obvious that abstractions on the level of details are necessary. Packet-level simulators model the P2P protocols with precise protocol headers and field structures, whereas message-level simulators only take into account the number and sizes of the

[+] The works of M. Vapa and A. Auvinen are supported by Graduate School in Electronics, Telecommunications and Automation (GETA).

packets. While packet-level simulation is a desirable feature, it is still often too expensive in terms of computing resources. In addition to speed, other desirable features in our simulations are compilation of statistics on simulation results and visualization of P2P networks. From this viewpoint, we next overview some of the existing P2P simulators.

A. NS-2

NS-2 [15] is one of the most widely used network simulators. The NS-2 is object-oriented discrete event simulator, which closely follows the architecture of the OSI model. It suits well for simulating packet switched networks and small scale networks. The Parallel and Distributed Simulation (PADS) research group has developed an extension that allows network simulation to be run in parallel on multiple machines [17]. Being very detailed simulator, it still does not scale well enough and is slow from a computational point of view. In addition, adding new modules is not straightforward, because of it's complex module structure [14].

B. PLP2P

Packet-level Peer-to-Peer Simulator (PLP2P) [6] provides a framework for other packet-level simulators, e.g. NS-2, in order to provide detailed model of the underlying network. This is done with wrappers, which translate P2P events into underlying packet-level simulator. The authors assert that abstracting low-level details can impact the simulation results to a large extent. The scalability problem of packet-level simulations is solved by running simulations on parallel machines. Nevertheless, as training neural networks requires substantial part of available computing power in order to get result within reasonable time, we need to abstract the level of details of the P2P network.

C. QueryCycle

The QueryCycle simulator [19] is specialized to file-sharing simulations. It has realistic models for content distribution, query activity, download behavior etc. The content distribution is based on a model, where each file belongs to one category and that category is defined by the popularity of the file. Simulations proceed in query cycles representing the time period between issuing a query and receiving a response. Generated queries are passed into a queue and handled on a First-In-First-Out basis.

D. 3LS

3LS [21] is an open-source simulator for overlay networks designed to overcome the problems of extensibility and usability. The system is separated to three architectural levels: a network model, a protocol model and a user model. The network model uses a two-dimensional matrix as a storage of distances between the nodes. The protocol model defines the current protocol being simulated. The user model is the input interface for the user. The 3LS uses most of the memory resources to a graphical interface as the simulator uses main memory to store each event executed for visualization and this limits the system to less than a thousand nodes on a low-cost workstation [14].

E. PeerSim

PeerSim [18] has been developed especially with scalability and support for dynamicity in mind. PeerSim is Java-based and has two simulation engines, one is cycle-based and the other is event driven. Cycle-based engine allows scalable simulation but is not very accurate. Handling large-scale overlay networks requires simplifying assumptions about the simulation details. For example, the details of the transport communication protocol stack are not taken into account. Event driven engine supports dynamicity and is more realistic, but decreases the scalability of the simulation. The abstractions of cycle-based simulations are similar to ours. The difference is that when PeerSim uses the benefits of abstraction for high scalability, we use it to increase the computational efficiency. Parallel execution is a necessity in order to PeerSim to be useful for our research.

F. NeuroGrid

The NeuroGrid simulator [8] was initially designed to support comparative resource search simulations between FreeNet [5], Gnutella [16] and NeuroGrid [8] systems. The simulator is single-threaded, Java-based and uses discrete events. Several protocols are now available for NeuroGrid e.g., Domain Name Service (DNS) and a distributed e-mail protocol. NeuroGrid supports property files that specify the parameters of the simulation to run. This includes the protocol to simulate, the parameters of the network and the amount of searches.

NeuroGrid would be a promising simulator for our research if it wasn't single-threaded and non-parallel.

G. GPS

The General Peer-to-Peer Simulator (GPS) [24] is aiming to respond to a call for extensible framework for simulating P2P networks efficiently and accurately. Efficiency is accomplished with message-level simulation instead of packet-level simulation. Improvement to the level of detail is achieved by tracking the network infrastructure and using a macroscopic mathematical model to obtain accurate estimate of the message behavior, e.g. TCP. The GPS also models downloads of the files, which is often left out from the simulators. GPS is extensible for modeling any P2P protocol, integration with a GUI and network visualization and provides support for topology generation tools.

The GPS is still in it's early stages and details about scalability, usability and performance are scarce. The GPS has also only been used for simulating BitTorrent, where resource discovery is not an essential problem. It is single-threaded, but

according to the authors their aim is to include multi-threading into the simulator in the future.

H. Summary

A comparison of the different characteristics for reviewed P2P simulators is shown in Table I. Overlay with Routers column tells if the simulator contains both the logical overlay network topology and the underlying router structure of the physical network. After surveying the existing literature about P2P simulators it is obvious that there is a need for standardization in the area of P2P simulation [7]. The field is highly fragmented and most of the current projects use their own simulators tailor-made for their purposes. One of the problems of the widely used simulators is their complexity and therefore poor scalability for P2P simulation purposes.

Although our project strongly supports the call for general open-source P2P simulator that is easily extensible and even some attempts for such a simulator have recently appeared, our research area is still too specified to be implemented satisfactorily in any other way than building a specifically optimized simulator. Training neural nets is computationally very demanding and requires parallel computing and simplified network simulation. To the best of our knowledge P2PRealm is the only message-level simulator that allows simulations on parallel machines. In addition, there is no other neural networks based P2P algorithms that we know of and neither simulators supporting neural network algorithms.

The problem of specifically built simulators is that the results are not exactly similar with the ones made by other simulators. This is a compromise that had to be made. In P2PRealm the most common P2P resource discovery algorithms are implemented to allow comparison with the ones neural networks create. This provides the baseline for results obtained in other simulators.

III. PEER-TO-PEER REALM

Peer-to-Peer Realm (P2PRealm) is a Java based peer-to-peer network simulator designed for optimizing neural networks used in P2P networks. The simulator has been developed in Cheese Factory peer-to-peer research project [4]. With the simulator, it is possible to determine a certain P2P network scenario and requirements for a resource discovery or topology management algorithm and get as an output a neural network optimized for that scenario. For example, a P2P network scenario could contain Gnutella's [16] topology, resource distribution and query pattern and the requirements could state that we want an algorithm, which needs to locate certain amount of resources (say 150) using as few query packets as possible. The end result would be an adapted resource discovery algorithm for that particular P2P network scenario. The first results of this kind of an algorithm development was reported in [23]. Also, the simulator contains implementations of various P2P resource discovery algorithms such as Breadth-First Search [13], Random Walker [12], Highest Degree Search [1,22] and optimal path K-Steiner Tree approximation [22]. These algorithms can be used as performance measures for neural network based algorithms or for studying their performance in different P2P network scenarios. The simulator has also been used for studying topology management algorithms for P2P networks [3].

The simulator is divided into four parts: P2P network, P2P algorithms, neural network optimization and input/output interface. P2P network contains the characteristics of a P2P network including the network topology, distribution of resources and query patterns of P2P network users. P2P algorithms contains the implementations of various resource discovery and topology management algorithms. Neural network optimization takes care of neural network structure and different optimization algorithms used for training the neural network structure. Input/output interface is used for reading configuration files and for outputting the statistics of training and final results. The final results consist of the optimized

TABLE I. CHARACTERISTICS OF THE CURRENT UNSTRUCTURED P2P NETWORK SIMULATORS

	Level of Detail	Parallel	Scalability	Overlay with Routers	Dynamic Network	Programming Language
NS-2	Packets	Yes	Very low	Yes	No	C++
PLP2P	Packets	Yes	Medium	-	-	C++
QueryCycle	Messages	No	?	Yes	Yes	Java
3LS	Messages?	No	Very low (<1000 peers)	Yes	?	Java
PeerSim	Messages	No	Very high (10^6 peers)	Yes	Yes	Java
NeuroGrid	Messages	No	High (300 000 peers)	No	Yes	Java
GPS	Messages	No	?	No	Yes	Java
P2PRealm	Messages	Yes	Medium (100 000 peers)	No	Yes	Java

neural network and the used query paths for different queries.

IV. Input and Output Interfaces

The following information is required as an input to P2PRealm (described in a configuration file):

- P2P network topologies containing the resource distribution
- Query pattern
- P2P resource discovery algorithm
- Percentage of available resource instances to be located in each query
- Number of queries executed in each training generation
- Neural network inputs
- Number of training generations, number of neural networks and the neuron structure of neural networks
- Optimization method

As an output Peer-to-Peer Realm (P2PRealm) provides the following files:

- The used topology and neighbor distribution
- A trace of training process with separate files for training and generalization sets
- The best and all neural networks of each generation
- Query routes started from each node of the P2P network
- Configuration file, which was given as an input

V. Training NeuroSearch

Next, we briefly describe how P2PRealm can be used for P2P algorithm development. As an example we use NeuroSearch resource discovery algorithm [23], but other algorithms for example topology management algorithms based on neural networks could be used [9].

NeuroSearch resource discovery algorithm uses local information about query situation in a peer-to-peer network to decide if query should be forwarded to a neighboring peer node or not. The local information can be e.g. number of hops the query has traveled, number of replies still needed to be located etc. The forwarding process is illustrated in the following algorithm:

1. *One peer node starts a query specifying a resource it wants to locate.*
2. *For each neighbor the node has, do the following:*
2.1 *Fill all the input fields of neural network.*
2.2 *Compute the output of neural network.*
2.3 *If output is greater than zero, then forward the query to neighbor and increase the number of sent query packets by one.*
3. *A forwarded query packet arrives to peer node. If this is the first query packet arriving to this node, check whether the peer node contains a resource being queried. If peer has the queried resource then increase the number of found resources by one.*
4. *Go to step 2.*

The algorithm terminates when there are no more query packets to process. At the end the quality of neural network is determined by the number of found resources and the number of query packets used.

To get good neural networks, they need to be trained so the algorithm has to be executed many times (typically millions query executions). There are various neural network weight adjusting algorithms and depending on the used methods the training times can vary a lot. Still, all optimization methods have in common iterative behavior and therefore executing the algorithm efficiently is an important feature of the simulator.

The internal execution loops of P2PRealm used for training NeuroSearch are illustrated in Fig. 1. Each simulation run can have multiple simulation cases, where each case has its own environment parameters according to the input information described in section 4. Furthermore, each case produces NeuroSearch resource discovery algorithm optimized to this environment accordingly. With multiple cases it is possible to do parameter sweeps and to eliminate the need of starting the simulator manually each time one wants to use multiple training environments. The execution of different cases can also be distributed on Peer-to-Peer Distributed Computing platform [11] further described in section 6.

Execution of one case is divided into three different sections:

- Training of the neural networks
- Analyzing the training of best neural network in generalization environment
- Analyzing routes of the best neural network after the training

First, the case has its P2P networks, neural networks and other parameters initialized. Then the simulator proceeds to the training phase. In each generation multiple neural networks are evaluated by forwarding queries according to the resource discovery algorithm presented above. The queries are forwarded in one or more P2P networks and statistics of the query performance of each neural network is recorded at the same time. Usually between generations it is worth to do more specific analysis of the best neural network in generalization

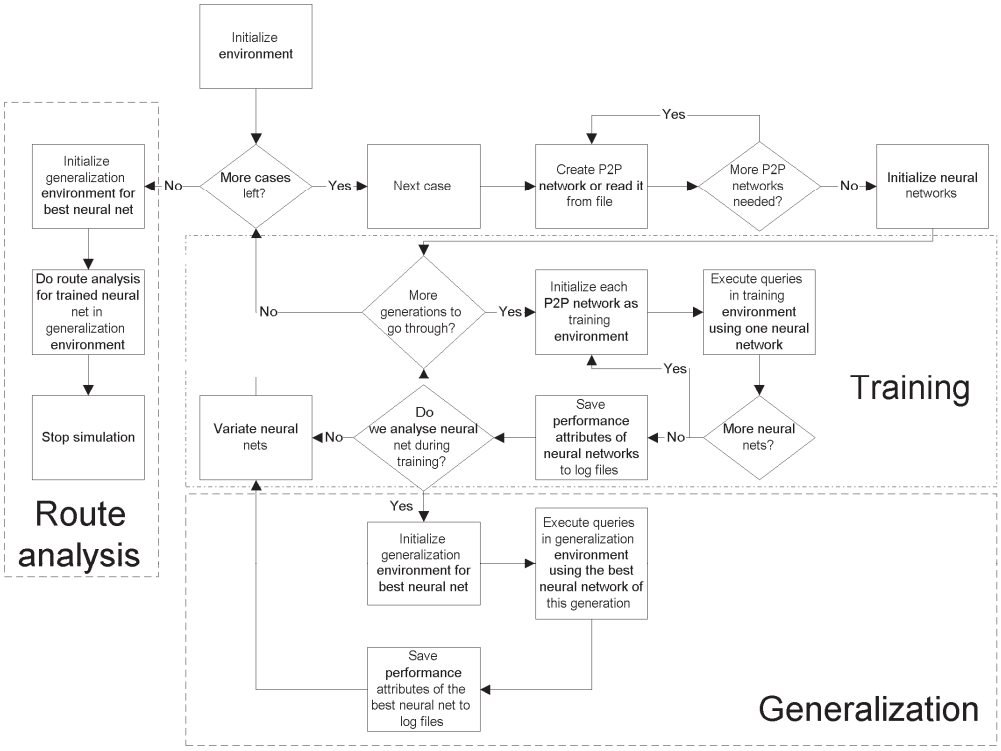

Figure 1. Execution Loops of P2PRealm

environment, where we can determine how the same neural network performs in an unknown environment. Generalization environment can be used to control when neural network is specializing too much on training environment and loses the ability to perform well in unknown but similar environments than the training environment.

After the evolution has proceeded the predetermined amount of generations, the simulator moves to the last phase of the process: route analysis. In the route analysis the same generalization environment is initialized as earlier for the best neural network, but now the queries start from each peer node at a time. The query paths produced by these queries are recorded and written to files to get accurate data about input and output values of neural network during a query. Finally, when routes have been recorded, the simulation ends.

VI. Speeding the Execution with P2PDisCo

The first implementation of P2PRealm used approximately one week for training the neural networks on a desktop computer. This was a severe limitation in research because it forced to study only small P2P networks and still getting results was very time consuming.

We started the internal code optimization process to see how much can be saved by optimizing internal loops of the simulator. After P2PRealm was profiled we found that the use of Vector object instead of Array in Java consumed lots of time (in particular getting the size of a vector through method call). Java container classes such as HashMap and HashTable can contain only objects and therefore reimplementing them to store only primitive values saved some execution time. Also we found that caching results of different method calls to avoid new method calls resulted in significantly faster execution times. The total time decreased to about 60% with these optimizations.

Java bytecode is interpreted in Java virtual machine yielding slower execution compared to compiled code. Java Native Interface [20] has been developed to allow native code for example compiled C++ to be executed from a Java program. We reimplemented the calculation of neural network output with C yielding an execution time about 70-80% compared to first version of P2PRealm. Combining both the internal code

optimization techniques and Java Native Interface implementation of neural network output calculation, we thus achieved execution time of about 50% compared to first version of P2PRealm.

This was however not enough, because reducing execution time of one simulation case from a week to 3-4 days was still quite slow. As a solution, we started developing Peer-to-Peer Distributed Computing platform (P2PDisCo) [11] allowing the distribution of simulation cases to multiple machines.

Earlier in our project [4] we had developed Chedar P2P middleware [2], which provided the basis for building P2PDisCo on top of it. P2PDisCo allows the workstations joined in a Chedar P2P network to publish certain distributed computing application as a resource in Chedar P2P network. When other Chedar nodes find this resource, it can be used to deliver needed input files to computing nodes and the produced output files to the node, which started the computations. For further information on the behavior of P2PDisCo the reader is referred to [11].

The speed up of execution with P2PDisCo is nearly linear, because each simulation case is delivered to different workstation. In university environment it is easy to locate machines, which are idle most of the time, so getting hundred of machines (and thus 100 times faster execution) was relatively easy scaling the research process to much faster rates. The resulting architecture is shown in the Fig. 2. Master denotes the peers, which create simulation cases and P2PRealm denotes the peers, which compute these cases.

VII. Visualization of Data using P2PStudio

Peer-to-Peer Studio (P2PStudio) [10] is a monitoring, controlling and visualization tool for P2P networks research. When combined with P2PRealm only visualization features can be used, because current version of simulator does not provide monitoring data during execution of a simulation. For visualization, P2PStudio provides functionalities to draw network topology and different graphs e.g., neighbor distribution of the topology. Also, the location of resources and query paths can be illustrated on a screen to qualitatively analyze how algorithms are performing. In case, that the simulation contains neural network, the input and output values of a certain query will be shown in a separate table. A screenshot of P2PStudio is shown in Fig. 3 and the specific features of P2PStudio are described in separate article [10].

VIII. Conclusions

Peer-to-Peer Realm is a simulator for studying P2P networks. Its unique functionalities contain training methods for neural networks and optimized speed of execution. By combining P2PRealm with other tools developed in our project, the simulator can grow to a large-scale distributed P2P research environment.

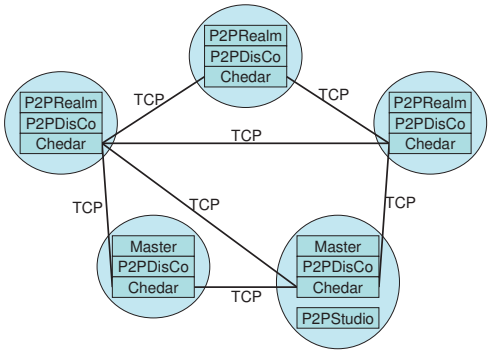

Figure 2. Architecture of P2PRealm combined with P2PDisCo, Chedar and P2PStudio

The future work of P2PRealm includes the parallelization of simulation such that multiple computers can process the same simulation task. Now only one simulation task can be allocated to a certain computer and speed ups are gained only when multiple cases are being simulated. Also, with the advent of multi-core processors for desktop machines, we are going to implement threaded version of simulator to support multiple processors within a single computer. For P2P network visualization, P2PStudio's user interface can be replaced in the future to support large P2P networks to be visualized and better usability of the program. Also the list of improvements for P2PRealm contain different query distributions and new input types for neural networks. As a longer term goal, we are aiming to combine neural network based topology management algorithms with neural network based resource discovery algorithms to study optimal construction of P2P networks.

References

[1] Adamic L., Lukose R. and Huberman B., Local Search in Unstructured Networks, *Handbook of Graphs and Networks: From the Genome to the Internet*, Wiley-VCH, 2003, 295-317.

[2] Auvinen A., Vapa M., Weber M., Kotilainen N. and Vuori J., "Chedar: Peer-to-Peer Middleware", Proceedings of the 20th IEEE International Parallel and Distributed Processing Symposium (IPDPS 2006), Rhodes Island, Greece, 2006.

[3] Auvinen A., Vapa M., Weber M., Kotilainen N. and Vuori J., "New Topology Management Algorithms for Unstructured Peer-to-Peer Networks", unpublished.

[4] Cheese Factory –project, http://tisu.it.jyu.fi/cheesefactory

[5] Clarke I., Sandberg O., Wiley B. and Hong T., "Freenet: A distributed anonymous information storage and retrieval service", *Proceedings of Workshop on Design Issues in Anonymity and Unobservability (ICSI),* Berkeley, CA, USA, 2000.

[6] He Q., Ammar M., Riley G., Raj H. and Fujimoto R., "Mapping Peer Behavior to Packet-level Details: A Framework for Packet-level Simulation of Peer-to-Peer Systems", *Proceedings of the 11th IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer Telecommunications Systems (MASCOTS 2003),* Orlando, USA, 2003.

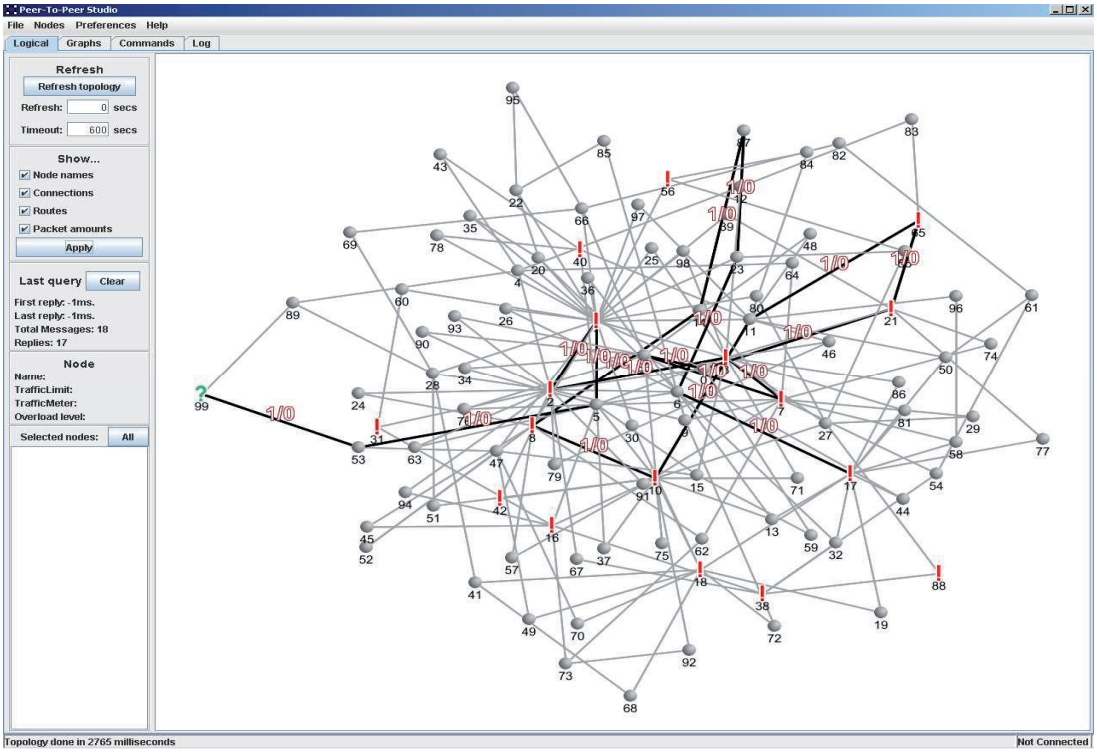

Figure 3. P2PStudio User Interface for P2PRealm

[7] Joseph S., "An Extendible Open Source P2P Simulator", *P2P Journal*, November 2003, 1-15.

[8] Joseph S. and Hoshiai T., "Decentralized Meta-Data Strategies: Effective Peer-to-Peer Search", *IEICE Transactions on Communications*, Vol.E86-B, No.6, 1740-1753.

[9] Keltanen T., "NeuroTopology: Topology Management Algorithm for P2P Networks", unpublished.

[10] Kotilainen N., Vapa M., Auvinen A., Weber M. and Vuori J., "P2PStudio - Monitoring, Controlling and Visualization Tool for Peer-to-Peer Networks Research", unpublished.

[11] Kotilainen N., Vapa M., Weber M., Töyrylä J. and Vuori J., "P2PDisCo - Java Distributed Computing for Workstations Using Chedar Peer-to-Peer Middleware", *Proceedings of the 19th IEEE International Parallel & Distributed Processing Symposium (IPDPS 2005)*, Denver, Colorado, USA, 2005.

[12] Lv Q., Cao P., Cohen E., Li K. and Shenker S., Search and Replication in Unstructured Peer-to-Peer Networks, *Proceedings of the 16th International Conference on Supercomputing*, ACM Press, 2002, 84-95.

[13] Lynch N. *Distributed Algorithms*, Morgan Kauffmann Publishers, 1996.

[14] Montresor A., Di Caro G. and Heegaard P., "Architecture of the Simulation Environment", Technical Report: D11, BISON project, University of Bologna, 2003.

[15] NS-2, http://www.isi.edu/nsnam/ns/

[16] Oram A., *Peer-to-Peer: Harnessing the Power of Disruptive Technologies*, O'Reilly Media, 2001.

[17] PDNS - Parallel/Distributed NS, http://www.cc.gatech.edu/computing/compass/pdns/

[18] PeerSim, http://peersim.sourceforge.net/

[19] Schlosser M., Condie T. and Kamvar S., "Simulating a P2P File-Sharing Network", *1st Workshop on Semantics in Grid and P2P Networks,* 2002.

[20] Sun Microsystems, Java Native Interface Specification, http://java.sun.com/j2se/1.5.0/docs/guide/jni/spec/jniTOC.html

[21] Ting N. and Deters R., "3LS - A Peer-to-Peer Network Simulator", *Proceedings of the 3rd International Conference on Peer-to-Peer Computing (P2P 2003)*, IEEE Press, 2003, 212-213.

[22] Vapa M., Auvinen A., Ivanchenko Y. Kotilainen N. and Vuori J., "Optimal Resource Discovery Paths of Gnutella2", unpublished.

[23] Vapa M., Kotilainen N., Auvinen A., Kainulainen H. and Vuori J., "Resource Discovery in P2P Networks Using Evolutionary Neural Networks", *International Conference on Advances in Intelligent Systems - Theory and Applications (AISTA 2004)*, Luxembourg, 2004.

[24] Yang W. and Abu-Ghazaleh N., "GPS: A General Peer-to-Peer Simulator and its Use for Modeling BitTorrent", *Proceedings of the 13th Annual Meeting of the IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS '05)*, Atlanta, USA, 2005.