P2PStudio – Monitoring, Controlling and Visualization Tool for Peer-to-Peer Networks Research

Niko Kotilainen, Mikko Vapa, Annemari Auvinen, Matthieu Weber, Jarkko Vuori
Department of Mathematical Information Technology
University of Jyväskylä, Finland
firstname.lastname@jyu.fi

ABSTRACT
Peer-to-Peer Studio has been developed as a monitoring, controlling and visualization tool for peer-to-peer networks. It uses a centralized architecture to gather events from a peer-to-peer network and can be used to visualize network topology and to send different commands to individual peer-to-peer nodes. The tool has been used with Chedar Peer-to-Peer network to study the behavior of different peer-to-peer resource discovery and topology management algorithms and for visualizing the results of NeuroSearch resource discovery algorithm produced by the Peer-to-Peer Realm network simulator. This paper presents the features, the architecture and the protocols of Peer-to-Peer Studio and the experience gained from using the tool for peer-to-peer networks research.

Categories and Subject Descriptors
C.4 [**Performance of Systems**]: Measurement techniques

General Terms: Measurement, Performance.

Keywords
peer-to-peer; P2PStudio; monitoring tool; research infrastructure.

1. INTRODUCTION
Peer-to-Peer (P2P) networks consist of a set of peer nodes. Each peer node makes decisions on where to connect and where to forward resource queries resulting in a complex self-organizing network. Studying how different algorithms are performing requires collecting data from the entire P2P network to obtain a global view. In P2P networks research people have used crawlers [5,9] to collect data locally available for some peer nodes. This approach however is only able to gather a portion of the P2P network's behavior, because some of the peers might not accept any new connections requested by the crawlers. Also, the crawlers can only gather information, which is accessible by the P2P protocol and thus they do not have direct means to control the peer's actions.

In our approach, we use a centralized server to contact peers in the P2P network and to set filters to the peers for what events the peers need to report back to the server. This allows measuring different properties from the P2P network extensively and globally. The graphical user interface presents the collected data visually thus making the interpretation easier compared to reading plain text log files. In contrast to crawlers, we note that our work is the first attempt to create a P2P research environment, which provides strict control mechanisms and accurate measurements for studying the behavior of different P2P algorithms.

To monitor the events of a P2P network a specific monitoring interface needs to be implemented in the peer nodes. This interface is used for setting different event logging options and for accepting incoming connections for data delivery from the centralized server. However, in presence of a large P2P network the centralized server can have lots of connections to manage and presents a potential performance bottleneck in our approach compared to local gathering of data done by crawlers. This architecture can however be scaled up by using multiple servers as is common in studies with crawlers [9].

The rest of the paper is organized as follows. Section 2 presents P2PStudio, its features, architecture and protocols. Section 3 describes how P2PStudio has been used in peer-to-peer networks research for studying the performance of peer-to-peer resource discovery and topology management algorithms. Conclusions and future work are discussed in Section 4.

2. PEER-TO-PEER STUDIO
The Cheese Factory –project [3] has implemented a Java-based peer-to-peer computing platform called Chedar [1]. Chedar can be used to build a network of workstations where each node provides and consumes resources such as computing power, files and devices. Currently, Chedar is used as a middleware for P2P Distributed Computing applications [7]. Chedar has also been extended to support mobile devices [8]. In order to test and monitor the Chedar network there was a need for a tool that enables to remotely control and monitor each peer and workstation in a centralized way. By executing the Guardian student project [4], the first version of Peer-to-Peer Studio was developed in 2002.

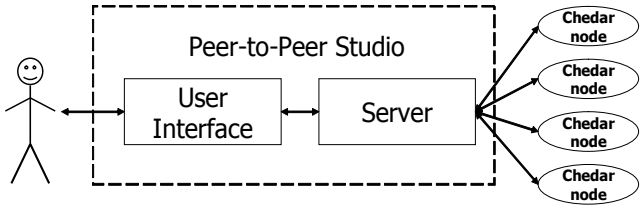

Figure 1. Components of Peer-to-Peer Studio.

P2PStudio is Java-based and it is divided into two separate programs as shown in Fig. 1: the user interface (UI) and the server. The graphical UI connects to the server program and uses

PM²HW²N'2006, October 6, 2006, Torremolinos, Malaga, Spain.

it to carry out the commands entered by the user. The server program takes care of all of the communication between the UI and Chedar nodes. It also manages the data sent from Chedar nodes. Dividing the application into two programs allows mobility of the UI from the dedicated hardware of the server. For example the server might have privileges to connect to Chedar nodes through firewalls and an UI residing on a laptop only needs to be able to connect to the server.

UI communicates with the server, sends requests to Chedar nodes, displays data from the server to the user e.g., by visualizing the network topology and showing diagrams. The UI also allows the management of Chedar nodes. Server forwards the commands sent by the UI, gathers information from the Chedar network and passes on requested data to the UI.

2.1 User Interface

The user interface draws a logical topology of the monitored network as shown in Fig. 2. From the zoomable topology view the user can select nodes and for example check their values, command queries to be sent and modify the resources owned by the nodes. Nodes can also be grouped together to ease the execution of a certain action to multiple nodes. Information on the last executed query is also shown in the topology view. The topology is generated using the WTS Veivi component from WTS Networks [12]. The component creates a visualization of network topology from a set of nodes and links optimized to minimum number of overlapping links. The topology is refreshed whenever the user desires or after a set interval.

Another feature of the UI is to show graphs of the monitoring data as shown in Fig. 3. Currently, the only graph implemented is the neighborhood distribution, but other graphs are relatively easy to be plugged in. Graphs are formed by combining multiple events into a single value, like in the neighborhood distribution, where individual neighbor amount notifications are counted and the frequency of certain value creates one data point in the graph. Graphs can be zoomed and shown also in a logarithmic scale.

The log feature of the UI allows the user to keep track of the Chedar network's actions almost in real time. Log presents the event messages coming from the Chedar nodes. The events are notifications of certain network events, for example forwarded queries, new neighbor connections or dropped messages because of congestion in a Chedar node.

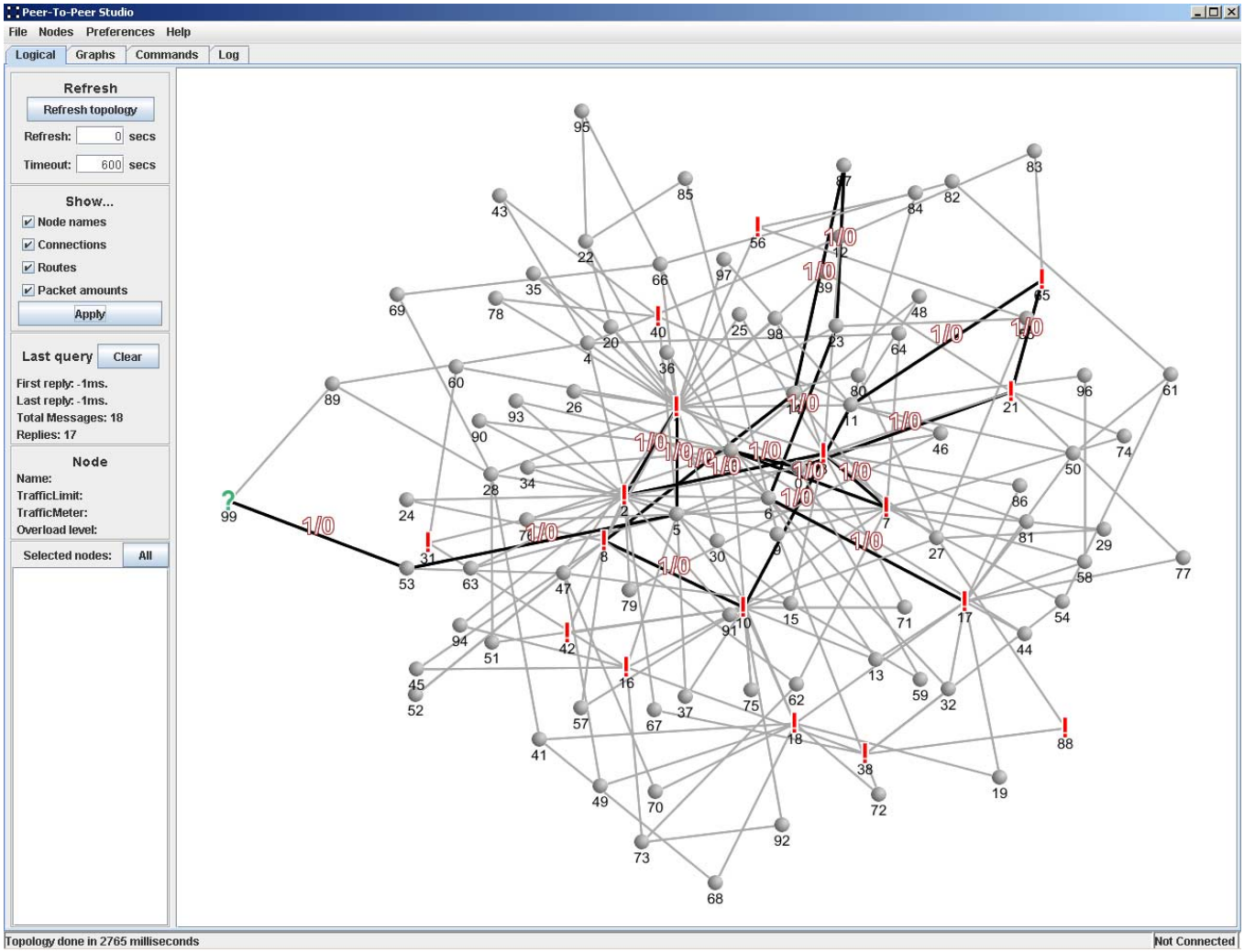

Figure 2. Topology view.

The user can also send commands to the server or to Chedar nodes via the server by typing commands in the User Interface-to-Server Message Protocol (UMP) format (for more details see the Section 2.3). The Commands view allows the user to see the sent data and the received messages from the nodes. Also batch files can be executed via the commands view. Batch files are useful when a certain peer-to-peer query pattern and measurement scenario needs to be executed multiple times.

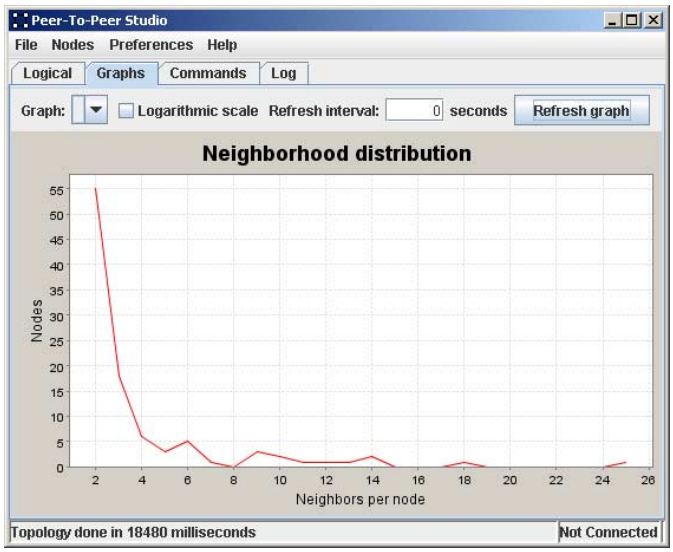

Figure 3. Graph view.

The UI can be run online as well as offline especially for demonstrations. For offline use there is a recording feature allowing the user to record actual monitoring data coming from the server to a file and later retrieve the recorded data in offline state. The UI also allows the user to create Chedar node groups and manage connections.

The functioning of the UI is quite simple. When data is received from the server it is checked and forwarded to the addressed component of the UI. The data will be presented to the user in a form of topology, graph or text depending on the view. Sending data is also rather straightforward. The user assigns a command and it is sent to the server for further handling.

2.2 Server

The server program is divided into two main components: stateless connection manager and stateful data manager. The connection manager is the part of the server which takes care of all connections. It forwards the contents of the packets without interpreting them, only adding metadata about the time the packet was received and Chedar node's IP address and port. A packet can arrive to the server either from the UI or from a Chedar node. It arrives first to the connection manager which forwards it to the data manager if necessary, otherwise directly to UI or to Chedar node(s).

The data manager is responsible for temporarily saving data coming from Chedar nodes and for combining multiple individual replies to a single reply for UI. For example to construct a neighbor distribution graph, data manager needs to collect individual neighbor amounts from Chedar nodes and build the graph data for UI. This lightweight architecture of the server allows scaling to hundreds of Chedar nodes.

2.3 Protocols

User Interface, Server and Chedar nodes use three different protocols for communication. One binary protocol was developed as a container for two message protocols, one XML protocol for communication between the server and the Chedar nodes as well as one XML protocol for communication between the UI and the server. Both XML protocols are on the top of the binary protocol as illustrated in Table 1. The binary protocol is always on the top of TCP.

Table 1. LAYERS OF THE PROTOCOLS.

Message Protocol (GMP or UMP)	*XML*
Packet Transmission Protocol (GPTP)	*Binary*
TCP	

1) Guardian Packet Transmission Protocol (GPTP)

The Guardian Packet Transmission Protocol (GPTP) is a binary protocol used between the UI and the server as well as between the server and the Chedar nodes. The GPTP packets are composed of a fixed-size 64-bit header and a data part, which varies in size. The header identifies the packet as a part of the Guardian-to-Chedar protocol and specifies the size of the data part in bytes. Without a specified data size, parsing an incoming XML message from a stream would be harder. An example of a GPTP message is shown in Table 2.

Table 2. GUARDIAN PACKET TRANSMISSION PROTOCOL.

32 bit synchronization header, 0x47324350 (G2CP)
32 bit size field, network byte order, (1234)
Byte data

2) Guardian Message Protocol (GMP)

The Guardian Message Protocol (GMP) is used between the server and the Chedar nodes on the top of the Guardian Packet Transmission Protocol. Each GMP message is a complete XML document. The header is a standard XML declaration, and the body is composed of a root element which specifies the type of message, and a variable content.

Here is the structure of GMP message:

Header: XML declaration

<?xml version="1.0" encoding="UTF-8"?>

Body

Root element: <request/> OR <reply/> OR <event/>

Content: various requests, replies or events as

XML elements and/or attributes

There are three types of messages in the Guardian Message Protocol:

Request	message is sent by the server to a Chedar or a Workstation node.
Reply	message is sent by a Chedar or a Workstation node to the server.
Event	message is sent by a Chedar node to the server.

The request/reply pair forms a synchronous message exchange initiated by the server. The reply is not mandatory. Event messages can arrive from the Chedar nodes at any time.

3) User Interface-to-Server Message Protocol (UMP)

The User Interface-to-Server Message Protocol (UMP) is used between the UI and the server on top of the Guardian Packet Transmission Protocol. UMP uses similar message structure as GMP. The difference between UMP and GMP is in the XML elements and attributes. For example the UMP contains elements for sending a certain GMP message to all Chedar nodes.

3. P2PSTUDIO IN PEER-TO-PEER NETWORKS RESEARCH

At first, P2PStudio was developed to collect data from a Chedar network [1] consisting of tens of workstations. Experimenting with self-organization of topology and different resource discovery algorithms however usually requires a controlled environment to obtain results that are repeatable. Creating exactly same starting conditions for each test in a network of workstations is problematic, because of differencies in hardware and network traffic. Also, having each Chedar node pack and send data over the network is significantly slower than executing algorithms in a simulator, where only local data structures are being used.

Therefore, the use of P2PStudio was extended by creating the Peer-to-Peer Realm (P2PRealm) network simulator [10,6]. P2PRealm is Java-based and contains functionalities for creating peer-to-peer network scenarios with different topologies, resource distributions and query patterns, executing different resource discovery and topology management algorithms, and collecting various statistics of the execution to log files. In addition to textual viewing of log files, P2PStudio can be used for graphical viewing e.g., to plot how queries spread in the network and what kind of topologies emerge from the execution of algorithms.

A special use case for P2PStudio and P2PRealm is the development of the NeuroSearch resource discovery algorithm [11], which is based on neural networks. Optimizing neural networks requires not only simulation of a certain scenario once, but usually thousands of times to reach a near-optimum state in learning. Therefore network simulators, such as Ns-2 [2], which are based on scripting languages and mainly developed for detailed protocol studies are not fast enough. For studying the behavior of neural networks, P2PStudio provides a view containing the inputs of neural network and the corresponding output decisions.

4. CONCLUSIONS AND FUTURE WORK

P2PStudio is a well-established research tool for peer-to-peer networks research providing functionalities for peer-to-peer network monitoring, controlling and visualization. P2PStudio has been used with two different peer-to-peer software, Chedar and P2PRealm, for algorithm development. The centralized architecture of P2PStudio is a potential bottleneck for scalability in the future when the size of the P2P networks being studied grows. As a future work we envision changes in the architecture to support multiple servers as well as adding new functionalities to UI to determine certain network characteristics such as diameter, shortest paths and multiple distinct paths between nodes.

5. ACKNOWLEDGMENTS

The authors would like to thank the other members of the Guardian student project: Joni Töyrylä, Jussi Rastas and Ville Pentti. Niko Kotilainen was supported by the InBCT-project and Mikko Vapa and Annemari Auvinen were supported by the GETA graduate school.

6. REFERENCES

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

[2] L. Breslau, D. Estrin, K. Fall, S. Floyd, J. Heidemann, A. Helmy, P. Huang, S. McCanne, K. Varadhan K. , X. Ya, and Y. Haobo, “Advances in network simulation”, IEEE Computer, Vol. 33, Issue 5, pp. 59-67, 2000.

[3] Cheese Factory – Peer-to-Peer Computing Project, tisu.it.jyu.fi/cheesefactory.

[4] Guardian project, www.mit.jyu.fi/opiskelu/sovellusprojektit/guardian/.

[5] M. A. Jovanovic, F. S. Annexstein, and K. A. Berman, “Scalability Issues in Large Peer-to-Peer Networks – A Case Study of Gnutella”, Technical report, University of Cincinnati, 2001.

[6] N. Kotilainen, M. Vapa, A. Auvinen, T. Keltanen, and J. Vuori, ”P2PRealm – Peer-to-Peer Network Simulator”, *Proceedings of the 11th International Workshop on Computer-Aided Modeling, Analysis and Design of Communication Links and Networks (CAMAD 2006)*, Italy, June 2006 .

[7] N. Kotilainen, M. Vapa, M. Weber, J. Töyrylä, and J. Vuori, "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.

[8] N. Kotilainen, M. Weber, M. Vapa, and J. Vuori, "Mobile Chedar - A Peer-to-Peer Middleware for Mobile Devices", *Workshops Proceedings of the Third IEEE Conference on Pervasive Computing and Communications (Percom 2005)*, pp. 86-90, Kauai Island, Hawaii, USA, 2005.

[9] D. Stutzbach, R. Rejaie, “Capturing Accurate Snapshots of the Gnutella Network”, *Proceedings of the 8th IEEE Global Internet Symposium*, Miami, Florida, 2005.

[10] J. Töyrylä, "Building NeuroSearch - Intelligent Evolutionary Search Algorithm For Peer-to-Peer Environment", Master's Thesis, University of Jyväskylä, 3.9.2004.

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

[12] WTS Networks, www.wts.fi