Towards a Visualization Tool for Peer-to-peer Overlay Networks

Fredrik Holmgren

SICS P.O. Box 1263, SE-164 29 Kista, SWEDEN
fredrikh@sics.se

January 25, 2005

SICS Technical Report T2005:01
ISSN 1100-3154
ISRN:SICS-T--2005/01-SE

Abstract

Studying how peers come and go in a model of a peer-to-peer overlay network and how it effects the distributed routing information can be a complex task. It usually gets to be a complex task even for a relative small network. Means for visualizing graphically the events and changes in an overlay network would be helpful for understanding, developing, verifying and demonstrating such networks. This report is the result of a short study towards designing and implementing such a visualization tool for the Chord protocol.

1. Introduction

1.1 Structured overlay network protocols and peer-to-peer systems

Peer-to-peer systems are distributed system without any centralized control or hierarchical organization. The key operation in a peer-to-peer system is efficient location of data items. To support efficient data lookup, structured overlay network protocols such as Chord [1] have been developed. Chord is a scalable protocol for data lookup in dynamic networks.

In Chord, peers are associated to identifiers on an identifier circle. A chord node/peer has routing information about only a few other nodes – a successor-list and a finger table. Thus the routing table is distributed and a node/peer resolves a lookup by communicating with a few other nodes. The finger table consists of pointers referring to id's located on logarithmically increasing intervals from the node/peer. Successor pointers give enough routing info but finger pointers offer logarithmic routing. When the network changes (peers come and go dynamically) Chord updates the networks routing information following a specific algorithm.

In the following text it is assumed that the term 'network' refers to a structured overlay network using the Chord protocol. The term 'network configuration' refers to a structured overlay network, implemented using the Chord protocol, at a given moment in time, with a specific set of peers and a specific set of routing information.

1.2 Motivation

Studying how a network and its distributed routing table evolves over time can be a quite a complex task. New peers join and leave the network, followed by subsequent (stabilization) processes that updates of finger- and successor-pointers. Already when studying small networks (with small id spaces) it becomes hard to keep track of the changes and their implications. In most situations the study would benefit from using some automated visualization tool, that may aid and structure the study and search among nodes and the myriad of correct and incorrect (not yet updated routing) links.

A tool for visualization for this task, would serve as a specialized interface between the user and a model/representation/simulation of the actual network. Such a tool would be helpful for a first time student of chord-based networks who would like to verify or extend his/her basic understanding. It would also be useful for a researcher that wants to refine the underlying algorithm(s) and would like to verify and track the effects of any alterations.

1.3 Visualization Scenarios

Network visualization scenarios may be classified according to their size in terms of their number of possible id's or by the mode of interaction. Large networks are the most challenging, but there are reasons for studying both large and small networks.

If the study is about specific details of the underlying algorithm and individual behavior of certain peers, then a small network may suffice and would also be more practical. If the study is about collecting statistics of the occurrence of individual and collective peer behavior a large network would be a more useful target.

If the collection of statistics is the prime goal, then it would be part of a simulation with a series of predefined or random events. In terms of interaction the user would (passively) monitor some collected and visualized data displaying the current state. However, if the primary goal is to study some individual events controlled by the underlying algorithm, it could be more practical to let the user (actively) interact with the system, invoking some events (actions). Such actions would be the explicit addition or removal of peers (under Chord) from an otherwise static network.

1.4 Development Approach

The system described in this report is the result of developing a visualization tool by starting from a simple system and then adding stepwise what is believed to be some useful extensions. The tool was developed within a short time frame and with the intention of possible inclusion into a larger simulation system – or alternatively just being an additional aid for researchers and students working with overlay networks. An overall purpose of the work was to get an initial understanding of what is required to simulate overlay networks and to get an understanding of Chord itself.

The initial starting point was that it should be possible to generate and display small static networks. Id spaces were supposed to vary around 4 to 24. An id space was to be displayed as circle of nodes inhabited by peers. With the links of the finger tables represented as arcs between peers.

To make links less obscured and more clearly identifiable, only the finger entries at a given index

(K) should be visible at a time. That is, given that K= 3, all 'start' and 'node' links at the third entry of all finger tables should be visible.

Given an id space of size N, with an actual number of nodes P, show the network, generated by all Peer.finger[K].node, where K is the (same) given index for all finger tables.

In figure 1 the same network is display four times. One time for each finger index of all nodes (peers). All the identifiers of the circle are associated to a peer. All displayed routing information is up to date.

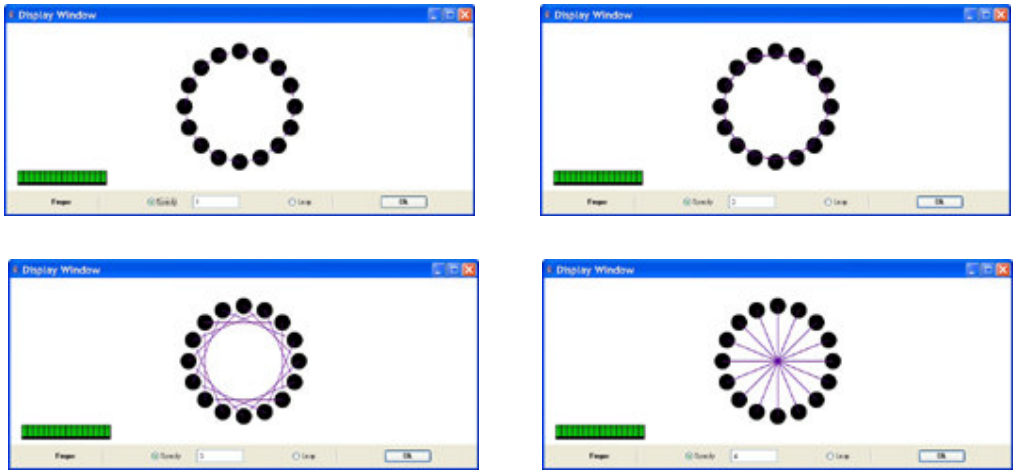

Figure 1: Four displays of the same network, each only showing a subset of its finger pointers.

In a second phase of the work, it was made possible for the user to interact with the otherwise static network to monitor the implications of modifying the network.

1.5 Structure of the Report

The rest of the report is structured as follows; First there is a section on the graphical user interface of the visualization tool which serves as an introduction to the section that describes the tools functionality. The implementation section gives a brief description of the tools architecture and implementation. The future work section describes ideas and directions for further development. The report is ended with a short summary.

2. Graphical User Interface

The graphical user interface (Fig 2) consists of two windows. A display window that shows the current network configuration and a control window that contains a set of buttons. In the display window two different views of the network is given. In the middle there is a circle depicting the identifier space. Each oval on the id circle represents an id. If an oval is filled black it means that a an active peer is associated to the id. Finger table entries are represented as arcs pointing from (the middle of) the peer having the link entry to (the perimeter of) the peer corresponding to the entry. 'node' links are solid blue while 'start' links are dashed red.

In the display window, to the lower left there is (linear) non-circular ordering of the id space. Here each active peer is represented by a staple of columns depicting its finger table. A low column

represents a low index and a high column a high index. A green column means that the 'node' link is correct whereas a red means that its not. An incorrect link is a link to an id whose owner (peer) is not longer active or is no longer responding (Fig 4).

At the bottom of the display window there is a parameter bar. The parameter bar is associated to the action being active and having been issued from the control window. Typically an action is chosen by pressing a button in the control window. Then the parameter bar for that particular command is shown ('New' in the figure above) and its name appears to the left in the parameter bar. After the correct parameters have been set the 'Ok' button to the right is pressed and the action is performed. To the left of the 'Ok' button there are two buttons 'bwd' (backward) and 'fwd' (forward) the lets the user browse between different (previously displayed) network configurations.

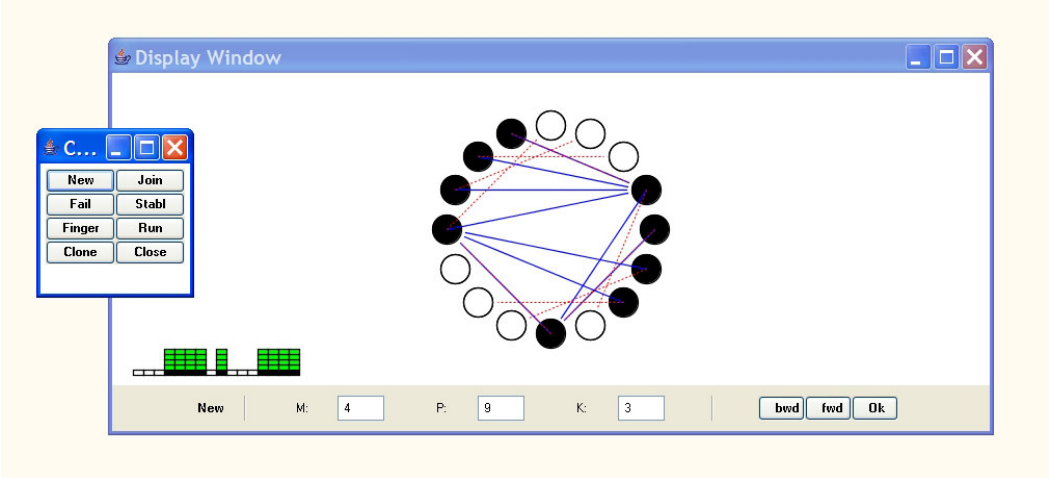

Figure 2: Graphical user interface

3. System Functionality

The actions that are invoked from the control window can be classified into three categories – network creation, network interaction and network display.

A creation action constructs a new network from a set of given parameters. An interaction corresponds to issuing a (simulation) events occurring in a network, like peers coming and going and the active peers updating their finger tables and successor lists. A display action sets what finger index to display or what network configuration to select from a history trace.

3.1 Network creation

'New' creates and displays a new network (fig 2). The size of the networks id space and finger table length (M, where M= $\log_2$ K and where K is the number of id's), the number of peers (P) and which finger (K) that should be displayed is given in the parameter bar.

'Clone' makes an independent copy of the network currently displayed and displays it in a new visualizer. A change in the original network does not affect the copy and a change in the copy does not affect the original.

3.2 Network interaction

'Join' adds a new peer to the network (fig 3). The id of the peer can be given explicitly or can be chosen randomly.

'Fail' removes a peer from the network (fig 4). If an explicit id is given, the peer with that id is removed. If no id is explicitly given a peer is chosen at random and removed.

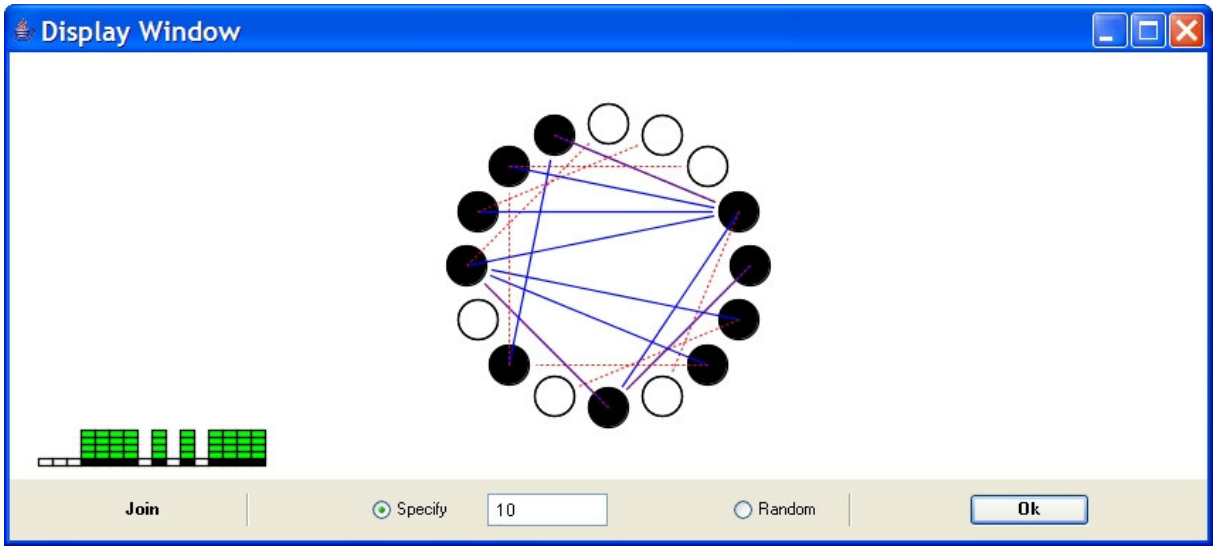

Figure 3: Join – an active peer has been added and associated to id '10' of the network shown in fig 2.

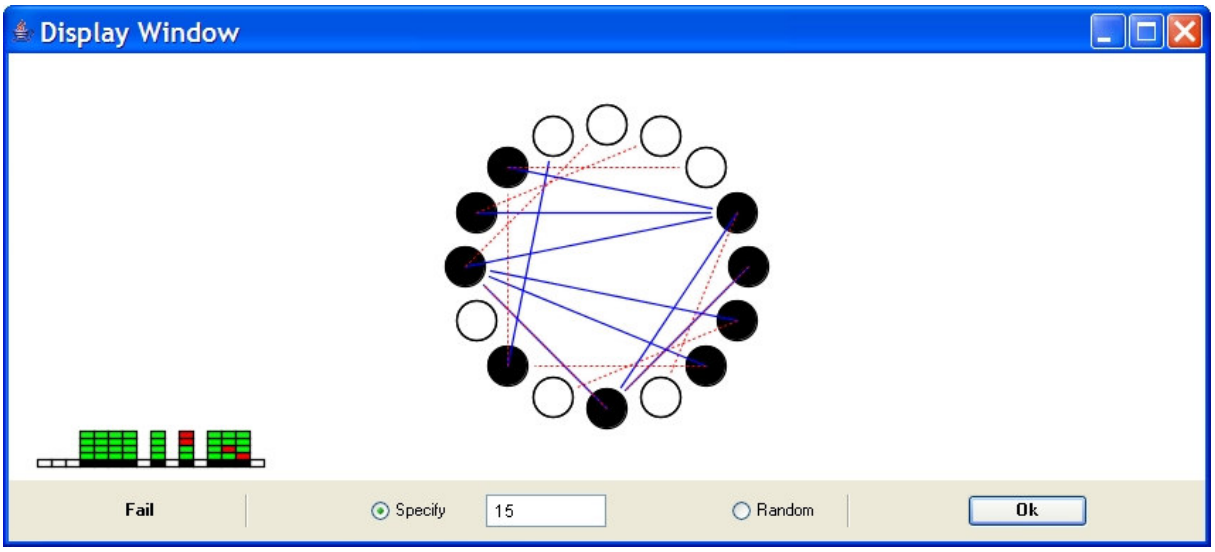

Figure 4: Fail – the peer associated to id '15' has been removed from the network shown fig 3.

'Stabl' runs the stabilization procedure of Chord randomly.

'Run' runs a mini-simulation of N steps where a random number of join, fail and stab actions are issued.

3.3 Network display

'finger' lets the user decide what finger-table links to display. An index is chosen and the 'start' and 'node' entries for all peers given that index will be visualized. Each entry will be represented as an arc from the peer having the link entry to the peer corresponding to the entry. 'Node' links are solid blue while 'start' links are dashed red.

In the current implementation the buttons 'bwd' and 'fwd' are found in the parameter bar of 'new'. They are situated to the left of the 'Ok' button. Each network configuration is stored internally in a history-list. The 'bwd' and 'fwd' commands let the user browse any previous configurations shown in the display window. If the configuration currently displayed is altered (a new configuration is generated) any history point following the currently displayed is truncated. A clone network does initially not have a history-list.

Figure 5 shows a clone of the network configuration shown in figure 4 (lower display). The other display has been browsed backwards two steps, to show the original configuration in relation to the clone.

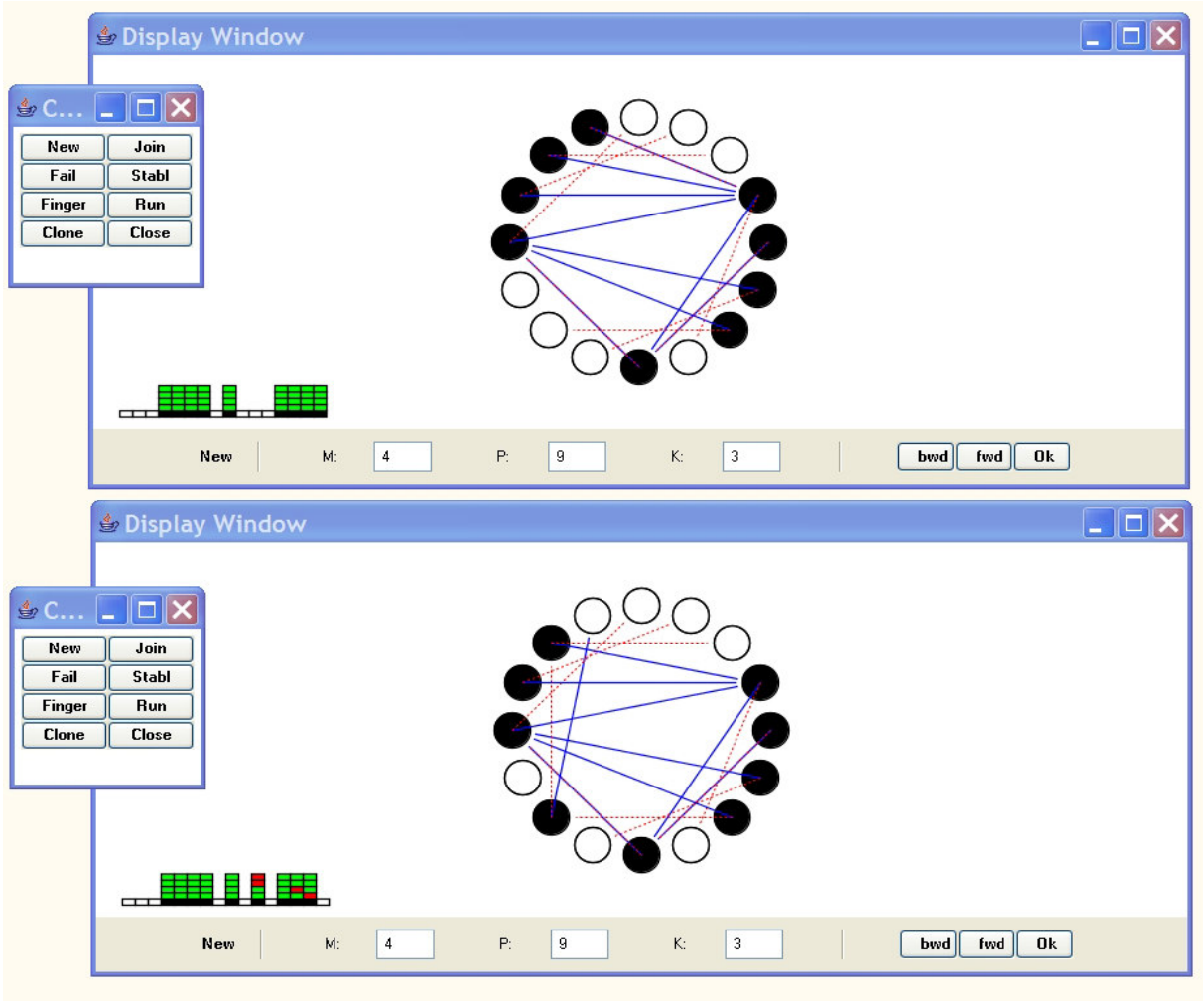

Figure 5: Cloned network

'Close' closes the associated display/visualization. Other active visualizations related by cloning are

left open.

4. Implementation

The visualization tool has been implemented in Java. Graphics are developed using the Java Swing and Graphics2D packages.

The Visualization tool in its present form consists of three parts. A graphical visualization system, a model or representation of the network and procedures representing the algorithms of the Chord framework and an interface between the two.

Since already a specialized network simulator had been developed at SICS [2, 3, 4, 5], its packages for network generation and modification were reused. These packages form the basis of the network representation and modification part. Only some minor changes were made including adding alternate constructor methods for network cloning.

A simple interface was designed on top of these packages for network extraction and external interaction with the network. The idea was the visualization system could be reused 'as is' using an alternate network package only having to rewrite the interface code.

The visualization system consists of two parts (packages) – visualization graphics (vGraphics) and a visualization engine (vEngine).

The graphics part consist code for the control window, parameter bar, display window and the drawing of the id circle and finger table.

The visualization engine is responsible for fetching data from the network and from the graphics and for keeping a local representation of the network being visualized. The local representation includes both network data and the corresponding graphical info such as placement.

The visualization engine maintains the history-list of network configurations for user browsing. When creating a new network, an object that represents the network and its possible interactions, is generated and returned by the network interface. The object is put on the history list and its network is visualized by the paint method of the visualization graphics. When the network is being modified by user interaction the visualization engine makes a copy of the network object and leaves the original network configuration object intact on the history list. After the copy has been changed – it is also put on the history list.

5. Future directions

The developed visualization system may be further extended depending on needs and potential usefulness. Any further development may be characterized in terms of smaller extensions (more development, less research) or as larger undertakings (more research) that requires revisions of the present work. Also, what part of the architecture that needs to be extended/modified and how that may affect its use of packages of, and possible inclusion in, a larger (simulation) system.

5.1 Smaller extensions

The smaller extensions would typically deal with practical things.

- Add the ability to only show 'start' or 'node' links.

- Ability to move nodes graphically (with the links still attached) to see links that has been obscured by other links

- Ability to select a peer graphically and to show its finger table and successor-list textually

- Introduce small pop-up windows identifying a link when the cursor points at it

- Associate a name to each network configuration (history point) and show it in the display window-

- Validity check of input parameters

- Ability to save a certain network configuration (history point) and possibly also its history for further use. Preferably into a format that can be read and understood by humans or into some graph format that can be printable. Ability reload a saved description into the visualization system.
An at runtime editable preference table, in which display preferences can be set (how to display links, peers...)

Larger undertakings would include exploring alternate abstractions and the animation of algorithmic detail

5.2 Alternate abstractions

The current visualization system relies on the network having a small number of nodes and the display of links of the same fixed finger index for all peers. A different approach (which would allow a somewhat larger amount of nodes) would be to only display groups of nodes. Preferably in combination with select finger index links. A set of nodes would reside inside a macro-node. Each node would consist of a set of sub nodes. There would be internal links between sub nodes and external between (macro-) nodes. The different levels would be traversable by the visualization tool. It should also be possibly to have different levels shown at the same time in different display windows.

A motive for exploring this direction would be the ability to view larger networks and to view the distribution of the number/ratio of correct/incorrect links. A question is how to choose the initial distribution of sub-nodes into macro-nodes.

An extension of this kind needs added functionality (not so much revision) to the visualization engine and the visualization graphics.

5.3 Animation of algorithmic detail

Using the present tool, only the result of how the network has changed is visualized. We never see how it changed in terms of a peer resolving the pointers in its finger table, performing a lookup as a part of another peer updating its finger table.

The question is how to visualize or animate peer behavior at lower algorithmic level in a way that can be easily comprehended. To illustrate network evolution and search in finer detail could be

useful for both education and verification of assumptions.

An extension of this kind needs added functionality to the visualization engine and the visualization graphics. It also requires additions and alterations to the classes representing the network itself.

6. Related Work

P2PKit [6] was developed as part of the EVERGROW project, is a decentralized service architecture, built on an overlay network backbone. The architecture allows to install, monitor and update peer-to-peer services. An example of a service is a monitor service named the Control Service.

In the programming model of the P2PKit, components can receive messages by subscribing to event streams. Examples of events are nodes joining and leaving and successor and predecessor change. The Control Center service uses such notifications to update its display as soon as any related events occur. The Control Service visualizes the state of each node in the network in real time. It has been used for visualizing large networks in the dynamic PlanetLab environment.

7. Summary

The work presented in this report is motivated by the fact that the study of how an overlay network changes over time can be quite a complex task. Visualizing generated networks and issuing events and to display the effect of those events using a visualization tool is a way to get a deeper understanding of overlay networks - which can help in both understanding and verification of such networks.

An initial visualization tool was developed during a short period of time. The goal was to investigate the potential usefulness of such a tool from the simple staring point of having a small number of nodes and adding some well defined extensions.

The visualization tool was implemented in Java and used parts of an existing simulator for overlay networks as the starting point. The added visualization part consists of a visualization engine and graphical user interface.

The visualization engine manages a description of both the networks topology and graphical representation of the current network. The user can invoke updates of the network from the graphical interface. Each updated network is collected as part of a history-list. The history list is maintained by the visualization engine and can be browsed by the user from graphical interface.

Future work may either focus on extensions making the present tool more practical or focus on alternative abstractions or animation of algorithmic detail. An interesting alternative abstraction would be to combine the selection of a fixed finger table index (for all peers) select a group of peers and treat them as one node. The present tool displays the changes of the network. Animation of algorithmic detail would not only show the changes but aim towards illustrating the lookup tasks that eventually lead to changes in the network.

8. Acknowledgments

The initial ideas on how to design a visualization tool for overlay networks came from discussions with both Supriya Krishnamurthy (SICS) and Sameh El-Ansary (SICS).

The work presented in this report was within the AMRAM project funded by the Swedish research agency VINNOVA.

9. References

[1]
Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications
Stoica I., Morris R., Karger D., Kaashoek M. F., Balakrishnan H,
ACM SIGCOMM 2001, San Diego, CA, August 2001, pp. 149-160.
http://pdos.lcs.mit.edu/chord/

[2]
A physics-inspired perfomance evaluation of a structured peer-to-peer overlay network,
Sameh El-Ansary, Erik Aurell, Per Brand and Seif Haridi,
The International Conference on Parallel and Distributed Computing and Networks (PDCN 2005).
http://www.sics.se/~sameh/pubs/

[3]
A physics-style approach to scalability of distributed systems,
Erik Aurell and Sameh El-Ansary,
LNCS Post-Proceedings of the Global Computing 2004 Workshop (Rovereto, Italy),
Springer-Verlag, 2004.
http://www.sics.se/~sameh/pubs/

[4]
Experience with a physics-style approach for the study of self properties in structured overlay networks,
Sameh El-Ansary, Erik Aurell, Per Brand and Seif Haridi,
Proceedings of the International Workshop on Self-star Properties in Complex Information Systems (Bertinoro, Italy), May 2004.
http://www.cs.unibo.it/self-star/papers/elansary.pdf

[5]
Theoretical and algorithmic aspects of sensor, ad hoc wireless and peer-to-peer networks,
Sameh El-Ansary and Seif Haridi.
An Overview of Structured Overlay Networks, CRC Press, 2004.
http://www.sics.se/~sameh/pubs/

[6]
Synthetic report of services, architecture and properties,
Section 5, EVERGROW Deliverable D.3a.2, January 7, 2005
http://www.evergrow.org