Knowledge Media Institute

Visualization of Dynamic Chat Communication

Ondrej Novak, Marc Eisenstadt, Pavel Slavik

KMI-TR-121

June, 2002

www.kmi.open.ac.uk/papers/kmi-tr-121.pdf

Visualization of Dynamic Chat Communication

Ondřej Novák[1], Marc Eisenstadt[2], Pavel Slavík[3]
[1]Czech Technical University, Department of Computer Science and Engineering
[2]The Open University, Knowledge Media Institute

Abstract

This work is aimed at visualizing the dynamic behaviour of very large communication networks. The visualization of large graphs and networks is a crucial part of many applications, for which typical approaches are too demanding of computational resources. Since one of the most important issues is a visualization of the dynamic behaviour of graphs, special aspects of the visualization of huge graphs with dynamic behaviour are discussed. The paper describes an algorithm that speeds up the visualization of very large graphs and provides fast access to underlying data structures. It provides an adaptive data structure for general dynamic graphs and discusses the adaptation of this structure to very large communication networks. Our method supports several special tools for efficient analysis and evaluation of the graphs representing communication networks.

Keywords: Dynamic graph, Dynamic visualization, Tree, Information visualization, Large graphs, Communication network

1 Problem definition

Distance learning and 'supported open learning' are key developments in modern educational systems. In contrast to conventional universities, where students receive books and other materials, attend physical classes and consult with tutors periodically, modern distance learning establishments use a mixture of conventional print materials, multimedia, and web-based media that enable students and tutors to communicate either entirely or at least partially via synchronous and asynchronous technologies that eliminate (or at least reduce) the need for face to face meetings.

For efficient use of distance learning applications it is very helpful to have a suite of administrative and evaluation systems, so that tutors can react more appropriately to students' requirements and questions. Such systems need to provide real-time information about the state of students and tutors and information about their ongoing communication. One of the best-known universities that specialize in distance learning is The Open University (http://www.open.ac.uk). The Open University has about 200 000 distance learning students throughout the world. We have been collaborating with The Open University's Knowledge Media Institute on creating efficient tools for better scoring and evaluation that will help their tutors in their work. Visualization of information is one of the best ways of introducing complex data to the user in order to be enable efficient evaluation.

The communication between tutors and students and between students themselves creates a dynamic communication network. From the abstract point of view the network is a dynamic graph, where users are nodes and messages or other data flowing between users are edges. The graph has several special aspects: it is very fragmented (see fig. 1), dynamic and huge. The high level of the graph fragmentation is caused by the large number of users of

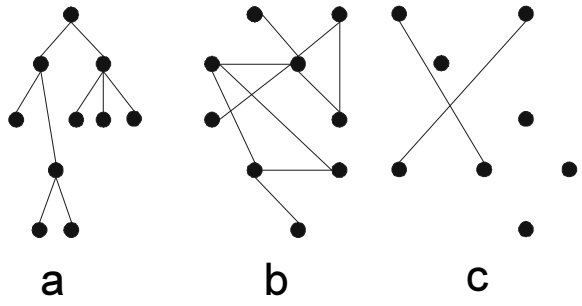

Figure 1: Types of graph:
a) tree graph, b) general graph, c) fragmented graph

which only small part communicates at one moment. This behaviour leads to a graph with many standalone nodes and few edges, whose position and state continually changes.

The most important aspect for visualization of dynamic graphs is their behaviour in time. It means continuous changes of node and edge numbers and changes of values of node and edge attributes. The aspect of easy-to-use navigation in the dynamic graphs is very challenging because of huge number of visualized objects, which are continually changing. The changes of data are one of the most important aspects to visualize in many applications. The user can clearly see which parts of data are changing

[1] xnovako2@fel.cvut.cz
[2] m.eisenstadt@open.ac.uk
[3] slavik@fel.cvut.cz

and which not and so he can easily recognize hidden information patterns. The changed data have to be highlighted in order to draw attention of the user.

The next important aspect is that however fast today computers are, continual change of large graphs is too demanding for their resources. Underlying temporal data structure can speed up work with huge graph. The structures are very useful with respect to use of focus and zoom method and filtering, because they reduce amount of searched objects. Since the graph behaviour is very dynamic, structure for fast updating is needful.

2 Context of our work

In information visualization, both navigation and representation are important. At first, the way that nodes and edges are displayed is essential, as this is a critical issue when the size of the graph increases. The techniques and tools for information representation provide an overview of the graph to the user and enable him to visualize specific node within some context. Fundamental factors for good visualization interface are: overview of the structure for a global understanding of the structure and of the relationships within, ability to zoom and to select some nodes and dynamic requests in order to filter data in real time.

Herman, Melancon, Marshall [6] point out that key issue for graph visualization is size of a graph, because the size can compromise performance or reach limits of viewing platform. They also show several problems with display cluttering that can make hard or impossible recognition of separate objects on display. Displaying an entire large graph may give only overall structure and helps to find out important locations but it makes difficult to comprehend details of graph without using other methods for the graph inspection.

Eick, Wills and Becker in [2], [5] describe basic rules for visualization of network data. They analyze communication networks with thousands of nodes and tens of thousands of links. They use both geographical and special layout algorithms for node distribution. Their visualization techniques involve static displays, interactive control and animation. Aggregation, averaging, thresholding and exception reporting methods were used to reduce amount of data. They named three main problems for effective visualization: display clutter, node positioning and perceptual tension.

T. Munzner [8] introduces two different methods for visualizing middle size and large size hierarchic networks. She uses hyperbolic view with zoom and focus method for improving navigation in large hierarchic networks in H3 project. The method is used for networks up to 100 000 nodes and the user can interactively inspect data from network. Next project visualizes MBone tunnel topology. It uses geographical layout and arc links in interactive 3D visualization in order to reduce display clutter.

Brown, McGregor and Braun [3] create tool for visualization of network performance called Cichlid. They use several visualization techniques based on animation for introduction of data flows inside network. They create abstract data model for network visualization and client server method is used for data collecting.

Donath, Karahailos and Viégas [4] present their tools for visualizing conversation that sustain from graphical interface for synchronous conversation – Chat Circles and program for visualizing threaded discussion - Loom. Their work is focused on highlighting social information and helping people make sense of the virtual world. They define new term – social visualization, which is defined as the visualization of social data for social purposes.

3 Description of algorithm

Since of these facts effective temporal structure was developed in order to speed-up visualization of large dynamic graphs with respect to previous aspects. The temporal structure is based on quad trees. The quad tree is based on layout attributes of graph nodes where nodes' attributes represent location of the graph node on the world map because every node is metaphor for one user, who is located somewhere in the world. Every graph node has got

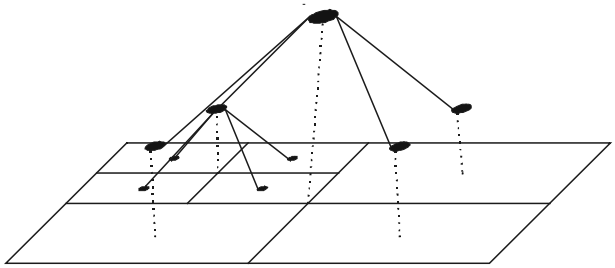

Figure 2: Tree built on top of cartographic layout

two layout attributes called longitude and latitude that describe its position on the world map. This layout is known as cartographic layout and is often used in cartographic information systems (fig. 2).

With reference to dynamic behaviour of the graph, the tree has to adapt to changing topology of the graph. The sub-tree that matched more close to the graph is built when any graph node is added to or removed from the tree. The adaptive quad tree is build in order to height of the tree should be minimal. By minimizing the height of the tree fast access to leaves of the tree is guaranteed. Since effectiveness of searching through the tree was considered, more than one graph node is stored in one tree leaf. Two limits for adding new graph nodes to the leaf are in the algorithm.

First one, soft limit denotes maximum number of graph nodes in leaf. If new node is added to the leaf with number of graph nodes in leaf equal to the soft limit, the leaf tries to create its new children and distribute its graph nodes to the children. If distribution is successful e.g. not all of graph nodes from the leaf are added to one child, the leaf is changed into inner tree node and new graph node is added to one of its children.

Second one, hard limit is usually two or three times greater then the soft one and denotes maximum number of the graph nodes in leaf. The leaf has to create its children

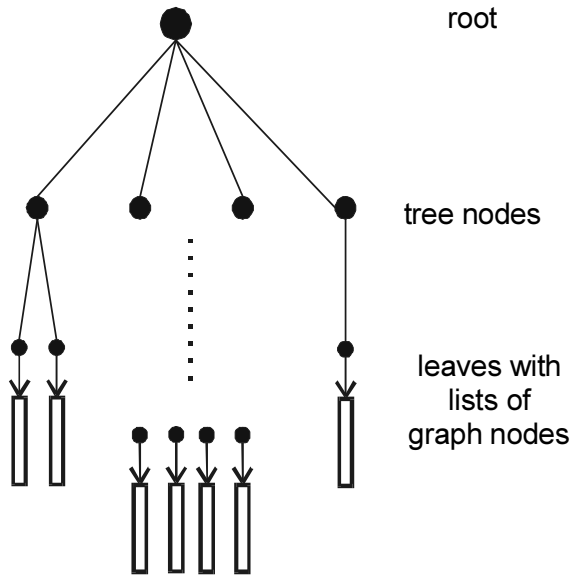

Figure 3: Structure of adaptive tree

tree nodes if its number of the graph nodes exceeds the hard limit. Both hard and soft limit depends on the number of graph nodes in the tree and is adapted to fit range of optimal proportion between global number of graph nodes and maximum graph nodes in one tree leaf (fig 3).

The efficiency of traversing the structure sustains from two parts. At first the tree has to be traversed till the leaf is reached. Then list of the nodes in the leaf is searched and nodes that match searching criteria are chosen. At some point is searching through short list of graph nodes faster than traversing longer tree structure. A typical example is the case where many graph nodes are very close. Long tree branch would have to be created. But traversing of that structure is slower than simple searching through the list of nodes. From these facts it is clear that number of the graph nodes in leaves should be adapted with respect to the global number of the graph nodes. The adaptation of maximum number of graph nodes in leaves of the tree balances time for traversing the tree structure and time for searching lists of the graph nodes in the leaves of the tree.

Since the graph is very dynamic, updating of the adaptive quad tree of nodes must be considered. Method of postponed updating of graph nodes was chosen in view of relatively low refresh rate of the user's display. The method is called update on demand. Every node has got structure of its changes. Changes are applied only if node has to be visualized on display. With reference to the adaptive quad tree every inner tree node has got a flag if any change has happened in its sub-tree. The flag speeds up traversing of the tree whenever only changed nodes are required. Before changed node is sent to visualization every changes from the list are applied. The update on demand method speeds up visualization, because no redundant data are visualized.

The attributes of the graph nodes and the tree nodes are very important for our algorithm. Every node of the graph has several important attributes: unique identification attribute, position attributes – longitude and latitude and other data attributes. The node of the tree has an attribute of change that indicates if some graph node lying in sub-tree of the tree node has been changed since last operation of taking the graph nodes from the sub-tree. Other attributes in the tree node are: attributes of rectangle covered by this tree node – longitude and latitude range, number of the graph nodes that lying in the sub-tree of the tree node and information about topology of the tree. The topology attributes are pointer to the parent tree node and vector of four pointers to the child tree nodes. The leaf nodes of the tree have list of the graph nodes instead of vector of the child tree nodes.

4 Implementation

The application is written in ANSI C++ using GUI library

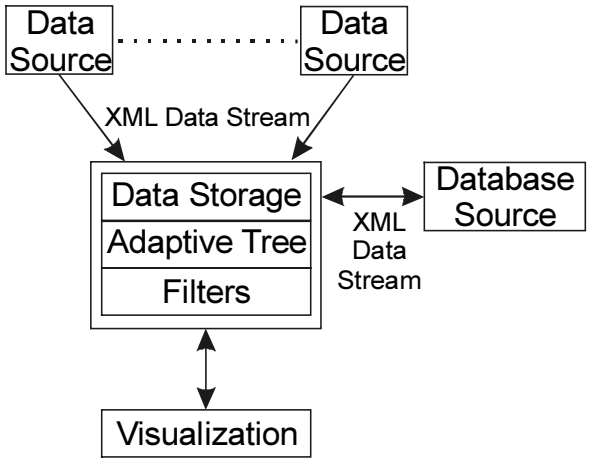

Figure 4: Architecture of the application

Qt [10]. It can be compiled both under Microsoft Windows and Linux/Unix platform. The architecture of the application is based on client server system. The application has three main parts: outer data sources, inner

data server module with adaptive tree engine and filtering engine and visualization client (fig. 4).
We use two main data sources. At first, they are communication data sources, from that we get information about communication activity between users. When the user log into the communication system, new graph node entity is created and after his logout his node is destroyed. The second source is a database server that gives us information about an user such as his position, name, etc. XML data stream is used for communication between data sources and data server. The stream is built on top of TCP/IP communication. The use of the XML data stream enables us simply extend current protocol in future and increase portability of the code.

The inner data server module contains three parts: main data storage, adaptive tree engine and filtering engine. The main data storage is a structure that stores all information about nodes and their changes. It provides fast access to every node by their unique identification key, for example username, etc. The adaptive tree engine stores information about distribution of nodes by their position. It is a hierarchic tree structure and it provides fast access to the nodes in all regions. It provides data for visualization client and filtering engine. The filtering engine presents an efficient way for the reduction of visualized information. The engine is open modular system that can be modified by the user. The user can add several modules for filtering nodes and edges of graph by their attributes and this engine can be used for clustering too.

The visualization client provides a graphic user interface for displaying graph, filters controlling and displaying information about selected object of the visualized graph. The interface enables the user to select diferent lenght of time period on that the graph represents state of the communication network, so that the user could recognize communication patterns over longer time period. Zoom and focus technique are used for an efficient navigation through the graph. The user sees information about global distribution of node clusters in one window. The information contains number of the nodes in cluster, information about change flag in cluster, e.g. if some nodes in the cluster have been changed from last displaying of the cluster. The user interactively selects some region and data from the region are filtered and are displayed in other window in detail. Any displayed object can be selected and the user can get all context information about the object in

$$t_c = t_{tree} + t_{leaves},$$

a separate information window.

5 Results and tests

Since the algorithm speeds up searching through huge data, we deduced theoretic time complexity of our algorithm. Total time for the searching using our structure is:

$$t_c \cong m * t_t * \log_4 \frac{n}{k} + m * k * t_s),$$

where t_{tree} is time for traversing tree structure and t_{leaves} is time for traversing list of the graph nodes in tree leaves. The height of the tree is:
where n is number of graph nodes stored in the tree and k is maximum number of graph nodes in one tree leaf, k is the number called soft limit in section 3.
For extracting of m graph nodes from the tree the maximal time t_c is equal to:
where t_t is time for traversing one level of the tree and t_s is time for extracting one graph node from the list in the leaf and comparing the graph node position with the searched area. For usual usage of the tree is lesser than t_c because given area is searched for graph nodes. In this case the tree is not traversed from the root to the leaf for every searched graph node, but only partial sub-trees are traversed. If all area covered by the leaf is inside the searched area, none of stored graph nodes in the list have to be compared with the searched area and whole leaf's list of graph nodes is

$$t_{sq} = n * t_s,$$

appended to the result set.

If only changed graph nodes are searched the time for searching is usually much lesser or at worst equal to t_c because only changed sub-trees are traversed. In the worst

$$t_c << t_{seq}, \text{ so}$$
$$t_t << c * t_s,$$

case, if some nodes are changed in all leaves, time for searching changed nodes is equal to t_c .

Sequential time t_{sq} necessary for searching for m graph nodes through list of graph nodes is:
where n is number of graph nodes stored in the list and t_s is time for extracting one graph node from the list and comparing the graph node position with searched area. The method is efficient if:
where c is constant.

The speed of our implementation of algorithm was tested. The tests were made on computer with processor PIII 600MHz and with 512MB memory. At first, we tested time that is needed for searching 1000 graph nodes in random area. All result times are measured in milliseconds. We tested for different datasets, it was graphs with 10 000, 50 000, 100 000 and 150 000 nodes. We tested both tree and sequential searching. The results of our tests are in table 1 and at figure 5.

Nodes in graph	Sequential time	Tree time
10000	146.643	2.997
50000	727.485	2.503
100000	1456.051	2.251
150000	2124.817	2.125

Table 1: Times [ms] for searching for 1000 graph nodes

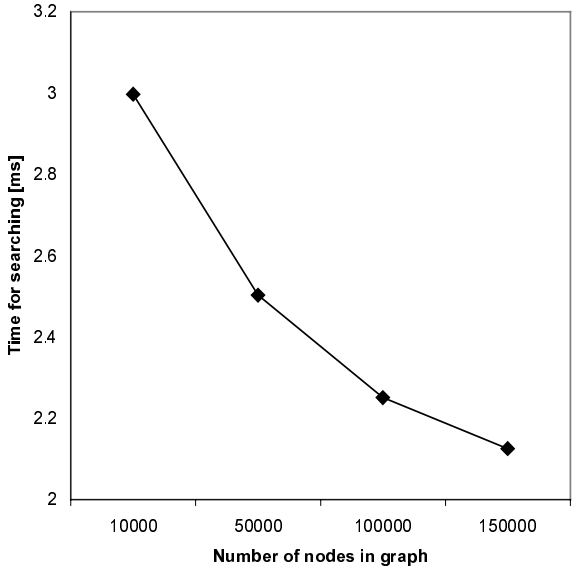

Figure 5 Time for tree searching for 1000 nodes, 10 nodes in leaf

Next we measured the time for searching 1000 graph nodes in trees with different distribution of graph nodes in tree leaves, e.g. with different soft limit. Results are in table 2 and at figure 6. From figure 6 is clear that as the time for searching as the ratio between searched and selected nodes depend on maximum number of graph node in leaf.

Results from the visualization client are at figure 7 and 8. Typical message pattern from small group of users shows high level of fragmentation of the communication graph. The figure 7 shows this state. Only few users communicate one to another at same time. The rectangle objects at the figure represent tutors whereas circular objects symbolize students. However clear is the communication pattern with the small group of users, with more users display become more cluttered. The figure 8 shows the communication pattern for medium group of

$$h \cong \log_4 \frac{n}{k},$$

user with about 500 people. The highlighting of changes

Max. graph nodes in leaf	Number of graph nodes			
	10 000	50 000	100 000	150 000
10	2.997	2.553	2.721	2.125
20	4.594	3.352	2.569	2.360
50	5.033	3.733	2.984	2.811
100	7.665	5.666	3.495	3.116
200	16.488	7.849	5.283	4.213
500	19.717	14.275	6.988	5.719
1000	26.362	16.229	8.171	6.946
2000		28.979	12.855	10.207
5000			22.301	18.239
10000			24.873	23.954
15000				34.675

Table2: Times [ms] for searching 1000 graph nodes with different distribution in tree leaves

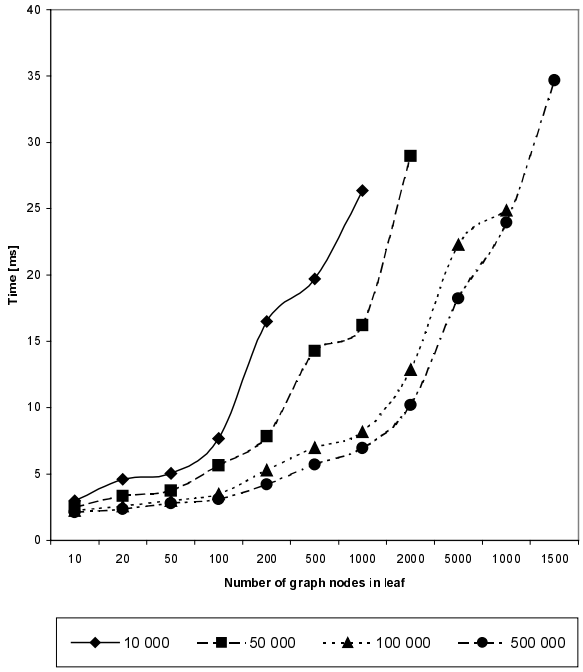

Figure 6 Time for tree searching, dependency on the number of graph nodes in a leaf

helps the user to fast recognize new objects in the communication graph and prevent him from fall into chaotic chunk of information.

6 Future work

There are many opportunities for making improvements to our tool. At first, we want to visualize longer time period and enable user to see communication patterns over longer time period. For the fast access to time data and history of

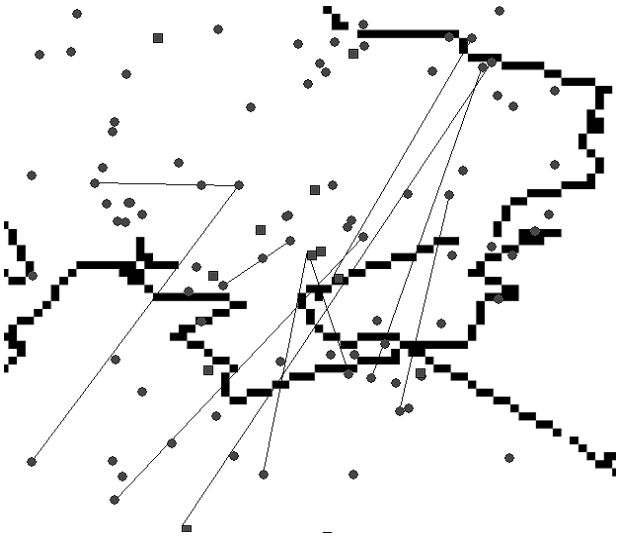

Figure 7 Communication pattern of small group of users

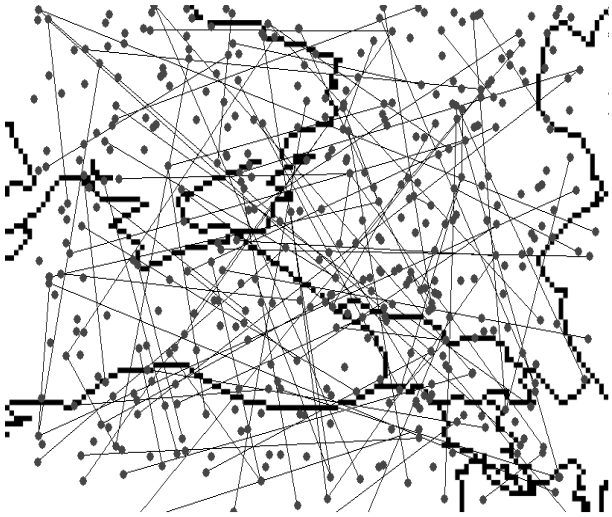

Figure 8 Communication pattern of medium group of users

every object in dynamic graph, we intend to create structure, that stores key events in "life" of the object and update events for the key events. The ability of evaluating longer period brings to the user powerful tool for evaluating new communication patterns. Then we would like to improve user interface of visualization client for easier interaction with the user. In future the client will be totally independent from main core of data server. Several new modules for working graph data will extend the core of data server. We would like to create a fast and platform independent communication protocol between visualization client and data server. The protocol should transfer demands of graph data, settings of filters and other modules and it will transfer graph data to the client. We are going to create thin version of the visualization client that will work with the web servers and it will create and send pictures rendered on the server side. The visualization client will only show the pictures and will send parameters for visualization to the server.

7 Conclusion

The visualization of large dynamic structures such as graphs and networks is a time consuming and not trivial problem. A method for speeding up the visualization was introduced. The algorithm is based on adaptive quad tree connected to update on demand visualization method. It speeds up the visualization of large dynamic graphs and brings a powerful tool for exploring such dynamic structures to the user. The measured results were discussed and effective speed up of visualization was noticed. The algorithm was tested on visualization dynamic communication network among users. Our application can visualize behaviour of dynamic communication network with thousands communication nodes. The system includes tools for filtering input data for easy recognition and exploration of hidden information patterns.

References

[1] O. Baudon, P. Auillans, *Graph Clustering for Very Large Topic Maps*, XML Europe 2001, 21-25 May 2001, Internationales Congress Centrum (ICC),Berlin, Germany

[2] R.A. Becker, S.G. Eick, A.R. Wilks, *Visualizing Network Data,* IEEE Transactions on Visualization and Computer Graphics, Vol. 1, No. 1, March 1995

[3] J.A. Brown, A.J. McGregor, H-W Braun, *Network Performance Visualization: Insight Through Animation,* PAM2000 Passive and Active Measurement Workshop, Hamilton, New Zealand, pp. 33-41, Apr. 2000

[4] J. Donath, K. Karahalios, F. Viégas, *Visualizing Conversation*, Proceedings of the Thirty-Second Annual Hawaii International Conference on Systems Sciences, January 1999

[5] S.G. Eick, *Aspects of Network Visualization*, Computer Graphics and Applications, Vol. 16, No.2, pages 69-72, March 1996

[6] I. Herman, G. Melancon, M.S. Marshall, *Graph Visualization and Navigation in Information Visualization: a Survey,* IEEE Transactions on Visualization and Computer Graphics, Vol. 6, No. 1, pp. 24-43, January-March 2000

[7] D. A. Keim, *Visual Exploration of Large Data Sets,* Communications of the ACM, August 2001/Vol. 44, No. 8, p.39-44

[8] H. Kurnar, C. Plaisant, M. Teittinen, B. Schneiderman, *Visual Information for Network Configuration*, University of Maryland CS technical reports, June 1994, Maryland USA

[9] T. Munzner, *Interactive Visualization of Large Graphs and Networks,* Ph.D. dissertation, Stanford University, June 2000

[10] http://www.trolltech.com