A Topology Viewer for Distributed Brokering Systems

Amey Dharurkar, Shrideep Pallickara and Geoffrey Fox
Community Grids Laboratory, Indiana University
(adharurk, spallick, gcf)@indiana.edu

Abstract

In this paper we present a topology viewer for the NaradaBrokering system. The NaradaBrokering system comprises a distributed network of cooperating broker nodes organized within a logical overlay network. The topology viewer's layout algorithm is implemented such that it augments the structure of the NaradaBrokering network. Furthermore, we believe that this approach could also be applied to similar visualization problems. Our visualization scheme extends recent work in the visualization of large hierarchies with primary focus on algorithmic and visual simplicity. Specifically, we have modified the ringed layout algorithm for placement of various NaradaBrokering components in order to achieve navigational efficiency. We also include performance measurements in this paper.

1. Introduction

Visualization of network topologies is a challenging problem. Networks can be thought of as graphs. Given a set of nodes and a corresponding set of edges, the basic graph drawing problem involves the calculation of the position of the nodes and the curve that needs to be drawn for each edge [5]. There are number of factors which decide the quality of a good drawing [6] of a graph. The size of a graph is a key issue in graph visualization [5]. As the number of elements in the graph increase, management of the layout becomes increasingly harder and compute-intensive, ultimately restricting system performance. Even if it is possible to layout and display all the elements, issues of viewability and usability persist. A typical layout algorithm may efficiently produce graphs with several hundred nodes but that does not guarantee that it would scale up to thousands of nodes with navigational efficiency.

At the highest level, graph layout techniques can be classified into *geographical* and *logical* layouts. Geographical layout techniques place each node on a map at its precise physical location, e.g. longitude and latitude, and then draw lines between these connected nodes. In 1997, CAIDA developed Map-Net [8], a tool for visualizing the infrastructure of multiple backbone providers simultaneously based on the geographical layout technique. Logical layout techniques [7] position graph elements based on their connectivity rather than their physical location. Such techniques reveal more information about the structure of a graph, its hierarchical nature, and its clustering factor. The design of logical layout techniques is more difficult than geographical techniques. Otter [8], a general purpose visualization tool developed by CAIDA incorporates logical as well as geographical layout.

When we explore logical layout in detail, it is often the case that the information to be visualized exhibits some form of hierarchical relationships. Further, it is often desirable to reduce the number of visible elements being viewed. Limiting the number of visual elements improves both the clarity and the performance of the layout [5]. Clustering is a key approach for achieving the abstraction necessary to reduce the visual complexity of a graph.

This paper is organized as follows. Section 2 presents an overview of related work in the visualization of hierarchical and clustered graphs. Section 3 provides a brief overview of the NaradaBrokering system. In section 4 we present the design of the topology viewer for visualization of the NaradaBrokering system. Section 5 presents results from our experiments. Finally, we outline the conclusions and future work that could be derived from our current work.

2. Related Work

Several techniques have been developed in order to visualize hierarchical structures. The fundamental problem in these techniques is how to provide enough details at lower levels in a hierarchy, while maintaining the context within the overall structure. Other desirable features of such techniques would include facilitating easier navigation and exploration through the information hierarchy while ensuring smooth transitions among views [9].

Treemap [10] is a space filling layout method developed at the University of Maryland. It maps a tree structure into nested rectangles, each representing a node.

Size and aspect ratio of the rectangular areas are controlled by the value associated with the node. Efforts have been made to make this space filling technique more effective in visualizing information hierarchy. Treemaps can easily handle hierarchies with tens and thousands of leaf nodes. Shneiderman and Wattenberg have suggested an algorithm for producing ordered treemaps [12] which improves stability in the display of dynamically changing data and preserves underlying order between tree elements. Treemaps efficiently utilize screen space by using 100% of available display and excellently portray leaf nodes but the technique is useful for fairly simple hierarchical plans [9].

Cone tree [13] is another simple technique which presents hierarchy in 3D. A cone is projected from parent node and its children are displayed along the circumference of the base of that cone. Cone tree adjusts the height of each level so that all cones at the same levels are of same height. There are some disadvantages in this technique. First, the display can only handle about 100 nodes or 10 layers of branching. Second, its 3D rendering is time consuming. Finally, a general problem with traditional techniques is that they are applicable for relatively small graphs, which makes them unsuitable for visualizing the NaradaBrokering network, a network that can expand up to thousands of nodes and links.

A practical solution is simply to layout a spanning tree for the graph, which forms the base for advanced tree layout techniques [5]. One such algorithm, presented by Lamping, Rao and Piroli [14, 15], takes advantage of the hyperbolic geometry. The H3 layout technique proposed by Munzner [16] is a hyperbolic layout technique which optimizes the cone tree layout algorithm by placing children nodes on the hemisphere around the cone instead of on its perimeter.

Clustered graph is yet another category of graphs which exhibits recursive clustering structure over the vertices [17]. In two dimensions, clustering structure is represented by region inclusions. Heuristic methods for such drawing have been developed by Sugiyama and Misue [18, 19], North [20] and Madden [21]. Straight line planar convex drawing of clustered graphs using Tutte's algorithm [22] is common. In orthogonal rectangular drawing [23] of a clustered graph, edges are drawn as sequences of horizontal and vertical segments while vertices are drawn on grid points and regions for clusters are represented by rectangles inside the grid diagram. As structure of a graph becomes more complex, 3D multilevel visualization techniques are used. All these 3D visualization techniques are time consuming in terms of navigation and rendering and thus do not fit into our visualization problem that aims at simplicity and clarity.

Circular layout or radial layout is commonly used for placement of clusters their sub-clusters and nodes. The graph layout toolkit developed by Tom Sawyer Software [24], includes a *circular* library whose layout algorithm places clusters on circles according to the logical interconnections of the clusters while ordering them to minimize crossings and also to reduce overall edge length. NicheWorks [25], visualization tool developed at Bell Laboratories uses radial tree layout as an initial layout for examining large telecommunication networks [26]. All of the above techniques which implement circular layout have a fundamental problem in that they inefficiently use the available space for the display of the hierarchical structure. **R**inged **I**nteractive-**N**avigation **G**raph **S**ystem (**RINGS**) [27] is a technique for visualizing large hierarchical data which overcomes the space utilization problem. The strength of RINGS is in its ability to show more area in focus and more contextual information than existing techniques while using a fixed resolution display. In RINGS a node is represented by a circle, however its children are placed as equal-sized circles in concentric rings around the center of the parent cluster instead of placing them on the circumference of the parent circle.

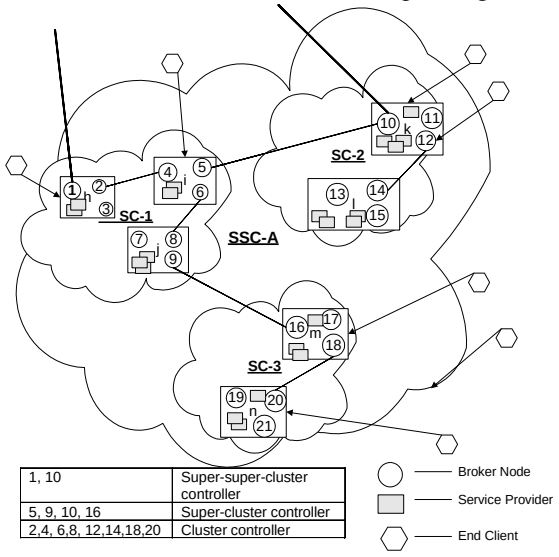
Children having large grandchildren are placed in the outermost ring inside the parent circle while the rest of the children are placed similarly in the inner rings. This kind of placement allows maximum space utilization. Further, RINGS allows the user to interactively explore large data. The primary focus is changed simply by clicking on the child to focus on. The child will be new center of the picture and the parent will be moved to the side. The algorithm thus maintains structural context even if the primary focus has been changed. Visual cues like color and transparency are frequently used to effectively enhance structural information.

In the design of Topology Viewer for the NaradaBrokering, we make use of such ringed layout with few changes in order to reduce visual and algorithmic complexity. We discuss our scheme in subsequent sections.

3. NaradaBrokering: A Brief Overview

NaradaBrokering [1, 2, 3] is an event brokering system designed to run on a large network of cooperating broker nodes. Communication within NaradaBrokering is asynchronous and the system can be used to support different interactions by encapsulating them in specialized events. NaradaBrokering efficiently routes any given message between the originators and registered consumers of the message in question. Messages could be used to encapsulate information pertaining to transactions, data interchange, system conditions and finally the search, discovery and subsequent sharing of resources. NaradaBrokering places no constraints either on the number, size or rate of these interactions. Scaling, availability and fault tolerance requirements entail that the

messaging infrastructure managing this information flow be based on a distributed network of cooperating nodes.



1, 10	Super-super-cluster controller
5, 9, 10, 16	Super-cluster controller
2,4, 6,8, 12,14,18,20	Cluster controller

Figure 1. Example of a NaradaBrokering broker network sub-section

The smallest unit of the messaging infrastructure that would provide a back bone for routing these messages needs to be able to intelligently process and route messages while working with multiple underlying network communication protocols. We refer to this unit as a *broker* where we avoid the use of the term *server*s to distinguish it clearly from the application servers that would be among the sources/sinks to messages processed within the system. Entities within the system utilize the broker network to effectively communicate and exchange data with each other. These interacting entities could be any combination of users, resources, services and proxies thereto. These are also sometimes referred to as clients.

To address the issues of scaling, load balancing and failure resiliency, NaradaBrokering is implemented on a network of cooperating brokers. In NaradaBrokering we impose a hierarchical structure on the broker network, where a broker is part of a cluster that is part of a super-cluster, which in turn is part of a super-super-cluster and so on. Figure 1 depicts a sub-system comprising of a super-super-cluster **SSC-A** with 3 super-clusters **SC-1**, **SC-2** and **SC-3** each of which has clusters that in turn are comprised of broker nodes. Clusters comprise strongly connected brokers with multiple links to brokers in other clusters, ensuring alternate communication routes during failures. This organization scheme results in "small world networks" where the average communication path-lengths between brokers increase logarithmically with geometric increases in network size, as opposed to exponential increases in uncontrolled settings.

This distributed cluster architecture allows NaradaBrokering to support large heterogeneous client configurations that scale to a very large size. Within every unit (cluster, super-cluster and so on), there is at least one unit-controller, which provides a gateway to nodes in other units. For example in figure 1, cluster controller node **20** provides a gateway to nodes in cluster **m**. Creation of broker network maps (BNMs) and the detection of network partitions are easily achieved in this topology.

4. The NaradaBrokering Topology Viewer

In this section we outline our design of the topology viewer for NaradaBrokering. We first outline certain objectives that we seek to achieve and then proceed to include a detailed discussion of our scheme.

4.1 Design Goals

In NaradaBrokering we limit the number of sub-units within a unit to 32. Thus, there can be 32 nodes, at most, within a cluster. In a 4-level system comprising of super-super-clusters we can possibly have 32x32x32x32 = 1,048,576 nodes within the system. Given the high number of nodes that can be part of the brokering networking, the layout should be as efficient as possible in order to minimize both algorithmic and visual complexity. Furthermore, the algorithm that is used for navigation should take negligible time as compared to the time required for redrawing the graph. This would ensure that the redrawing time remains the only deciding factor in navigational operations.

The layout algorithm should also quickly respond to modification (addition and deletion of nodes and links) to the underlying network fabric with minimum changes in the layout. The nodes in the graph should be evenly distributed over the available space ensuring that overall structure is easily visualized. Each level within the network also needs to be easily discernible.

Finally, it should be possible for users to concentrate on any small part of the broker network. In addition to this a user should be able to retrieve information associated with any particular node or link easily.

4.2 The Layout Algorithm

4.2.1 Node Placement. In our scheme we use a ringed layout algorithm where the parent node is represented by a circle as described in RINGS technique and all the children are distributed inside the parent circle along concentric rings located at half of the original parent-circle radius. However, as opposed to RINGS where the children are distributed over multiple concentric rings, we place all the children on the same ring. This reduces the number of calculations that need to be performed while plotting the graph on screen. This variation introduces the

disadvantage of reduced efficiency in space utilization but it improves the performance of the layout algorithm. The algorithm for distribution of child nodes in a parent node is propagated to the lowest level in NaradaBrokering network.

Mathematically, if there are *n* children to be placed inside a parent node then centers of circles representing the children coincides with the vertices of a regular *n*-angled polygon located in the middle of a parent circle. Angular distance between child nodes will thus be *2Π / n*. Length of the side of such a regular polygon can be easily calculated as depicted in the figure 2 below. The radius of the circle representing the child node is set to slightly less than **R Sin (Θ/2)** where R is the radius of the circle representing the parent node and Θ is the angular distance between the centers of consecutive circles representing the children nodes.

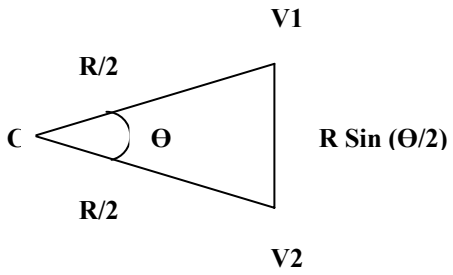


Figure 2. Calculation of Radius of a Child Node inside Parent Node

The node placement algorithm mainly aims at reducing the plotting time when a user views inner components of the graph. Since each component is represented by a circle and encompasses all its children, it becomes easy to determine whether a particular component can lie on the screen. For example, if a user is exploring a particular super-super cluster (after zoom operations) some of the other super-super clusters may not be seen on the screen. Mathematically, knowing the center and radius, it is simple to determine if the circle lies on a given rectangular area. If any circular component does not lie on the rectangular screen then it and all its children are simply excluded from drawing the screen image. This feature helps reduce the time required for redrawing the screen while a user views inner components, which is often the case.

Our design also provides a facility, using which, a user can see all the information related to a node (address, connections and supported transport protocols) by clicking right button of the mouse on that node. From the [*x, y*] Cartesian co-ordinates of the clicked point, we determine the node closest to that point. Due to inherent abstraction in the node placement algorithm, after a few comparisons we determine the closest super-super cluster, then closest super cluster inside the super-super cluster, and so on. Once the clicked node is found, the information associated with it is retrieved. Since the node placement algorithm reduces the number of operations that need to be performed, the response time associated with the retrieval of information associated with node is reduced.

We also introduce the idea of a virtual node, which does not exist physically but is the topmost level which contains all the super-super-clusters. At the start, the center of our virtual node is the center of the view port while its radius is half of the minimum between width and height of the view port. This virtual node is never drawn on the screen but its center and radius controls the locations of all the other components within the network.

4.2.2 Edge Management. Radial edges between nodes at adjacent levels, representing the tree structure as suggested in RINGS technique are avoided. Edges between broker nodes are drawn as straight lines. Uniform circular layout minimizes the overlapping of the edges. Additional edges between nodes at different levels can be shown in different colors so that the user can judge the interconnections easily.

4.3 Interactive Navigation

4.3.1 Interactive Navigation and Translations. Like other visualization techniques our design allows users to interactively explore the large hierarchical structure. It is also important that a user can navigate at a particular level by changing the focus. Change of focus can be achieved by simply clicking on that point in the graph. The clicked point moves to the center of the screen while rest of the scene within the view port is just translated to occupy new position.

For achieving this operation, the center of the virtual node is first shifted by the horizontal and virtual difference between old focus and new focus. The radius of the virtual node is unchanged. From the new location of the center of virtual node, positions of nodes at all the levels are determined and the graph is redrawn. Since every node is a circle, only two parameters need to be changed – the *(x, y)* coordinate of their centers. This translational operation is differs from the change of focus operation described on RINGS in that the RINGS approach maintains the entire drawing context.

This approach improves plotting performance. When a user is observing a detailed view at the lowest levels (after zooming) we present only that part of the graph on the screen. The choice of a circle for representing a component helps reduce the time required for changing parameters as only center of the coordinates need to be

changed. Figure 3 demonstrate this change of focus that we described.

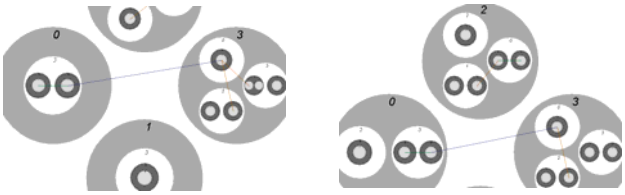


Figure 3. Views before and after the change of the focus

4.3.2 Zooming Operations. Zoom-in and zoom-out occurs at the focus (center of the screen). Zoom-in provides a more detailed view of the portion of the graph around the current focus. Consider that the current focus (screen center) is $[X_c, Y_c]$ while center of the virtual node is $[X_v, Y_v]$. When the zoom-in option is selected, the center of the virtual node is shifted depending on the zoom-in factor. Assume that zoom-in factor is 2. The radius of the virtual node is doubled while the new location of its center becomes $[(X_v – (X_c - X_v)), (Y_v – (Y_c - Y_v))]$. In general the formula for zoom-in factor N is as follows:

(a) $X_{vnew} = X_v – (N -1) (X_c - X_v)$

(b) $Y_{vnew} = Y_v – (N -1) (Y_c - Y_v)$

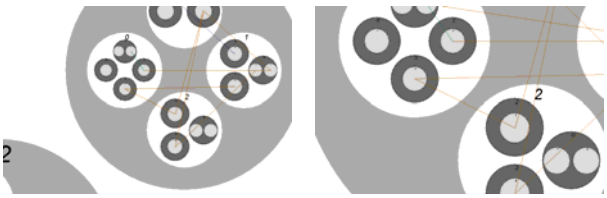


Figure 4. Views before and after Zoom-In

Once radius of the virtual node is changed and its new center is determined, centers of nodes at all the levels are recalculated using the basic layout algorithm while their radii are multiplied by the zoom-in factor. Due to the symmetric nature of the node placement, the above changes indeed reflect in perfect zooming around the screen center. Similar approach is applied for achieving zoom-out operation. This is depicted in figure 4.

4.4 Features of our approach

The whole graph is symmetrically distributed so the top view gives a good idea about underlying clustering.

The approach is very suitable for adaptive networks. Changes in any of the super-super-clusters does not affect placement inside other super-super-clusters.

The same algorithm is used for placement of child nodes within a parent node at any adjacent levels. Thus it is simple and efficient.

Navigational operations themselves take negligible amount of time and plotting of the graph is the only deciding factor. Drawing time reduces drastically as user observes detailed view at lowest levels.

5. Performance Measurements

In this section we discuss the results from our performance measurements. Plotting delay is the most crucial factor that determines smoothness between transitions. All the measurements were performed on a P4 (1.48GHz, 512 MB RAM, Windows XP) machine keeping maximum limit on number of children inside a parent node as 12 (i.e. maximum number of nodes in the system are 12 * 12 * 12 *12). Among the factors that we measured include graph plotting time, translation time and time for zooming operations.

5.1 Plotting a Graph

Drawing of a graphical image on to the available screen takes the maximum amount of time and is the sole factor which decides how smooth the transitions are. Assuming that the screen is size fixed, the top view of the topology lies inside the boundary of the virtual circle irrespective of number of nodes or links in the network. While plotting the whole graph, all the nodes, clusters, super-clusters and super-super-clusters are drawn, each representing a circle filled with some color. As number of nodes in the network increase, circles representing them become smaller and smaller due to the limited space. Obviously the time required to draw smaller filled-circles is less compared to time required for drawing bigger filled-circles.

It was observed that time required to plot the whole graph does not increase linearly with number of nodes. Furthermore, the time required to plot required portion of a graph decreases considerably as users are restricted to lower levels in the network. This phenomenon is explained in Section 4.2.1. Degree of independence from the number of nodes in the network is more prominent in such partial views. Figure 5 represents the time required to draw various components in the graph against a number of nodes in the system. The graph in Figure 5 clearly shows that as the view becomes more restricted, the time required to draw the image reduces. Furthermore, for restricted views, the plotting time increases less rapidly as the number of nodes in the system increase.

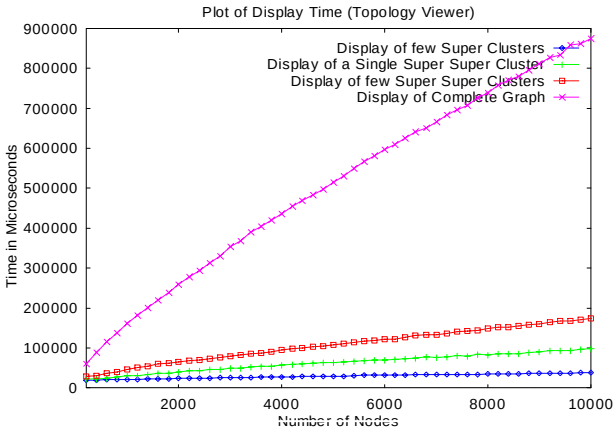


Figure 5. Time Required to Display Various Components

5.2 Zooming Operations

The time required for all the recalculation as described in Section 4.3.2, is in tens of microseconds which is far less compared to time required for drawing the image on screen. Time required doing such parametric manipulations increases linearly with the total number of nodes in the system as depicted by Figure 6**.** Complexity of redrawing the graph is identical to the explanation outlined in Section 5.1.

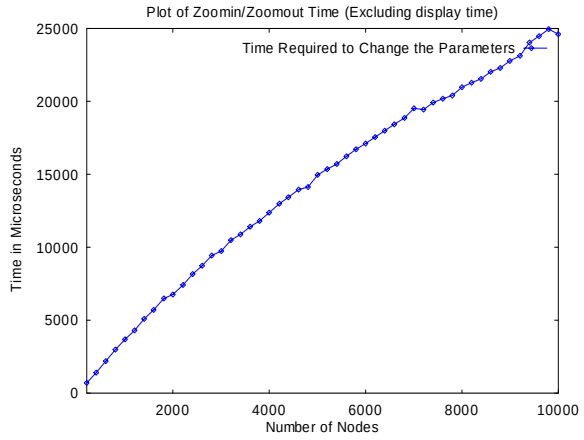


Figure 6. Time Required for Zooming

5.3 Translation Response

Whenever mouse is clicked over a point on the screen, whole image is translated so that the clicked point becomes a new focus and appears at center of the screen. As explained in section 4.3.1, it was observed that time required for changing these parameters is in few tens of milliseconds even if number of nodes exceed 10000. Once again most of the time is taken to redraw the graph.

5.4 Analysis of Node Operations

In our implementation of the topology viewer, the virtual node maintains a list of super-super-clusters. Each super-super-cluster maintains a list of super-clusters and so on. Given a node address, first the super-super-cluster id is compared with available super-super-clusters till a match is found. The process of comparison propagates to the lowest level (4 – corresponding to a broker node), till the required node having the correct node-id is found. Maximum number of comparisons are $4 * n$ where n is the maximum number of children per parent. Thus complexity of finding a node given its node address is $O(n)$. Complexity of adding a node or deleting a node also comes out to be $O(n)$.

Our implementation also provides a facility with which a user can see all the information related to a node (address, connections etc.) by performing a right-mouse-click operation on that node. Once again the maximum number of comparisons are $4*n$ and complexity of locating a node given $[x, y]$ coordinates is $O(n)$ where n is the maximum number of children per parent. Thus, the response time for locating a node is almost independent of the total number of nodes in the system. Graph depicted in Figure 7 illustrates that the response time (for getting node information) vary about certain interval once the network is populated by the nodes.

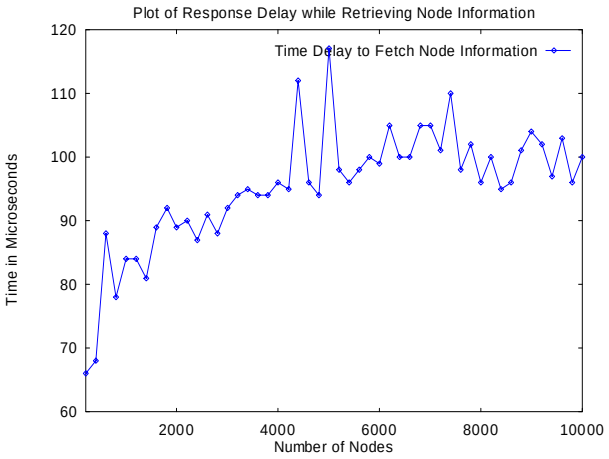


Figure 7. Variation in Response

6. Conclusions and Future Work

In this paper we have described the design of a topology viewer for NaradaBrokering. Our scheme is a variant of the RINGS approach. The scheme we outlined

is simple and very efficient. Though the design of the topology viewer is currently specific to NaradaBrokering we believe that it does have the underpinnings to be applicable to large scale messaging infrastructures. We have tested our approach for large broker networks and it can handle display of over one hundred thousand nodes. A simulation of the NaradaBrokering topology viewer can be found at [4].

NaradaBrokering incorporates a monitoring service at each broker node. These monitoring services monitor links that originate from the broker. We intend to augment the topology viewer with functions to access the monitoring service running at individual broker nodes to retrieve this performance information. Finally, we also wish to extend the topology viewer to enable individual users to run their own instances of the topology viewer which would coordinate and synchronize with the main instance of the topology viewer.

7. References

[1] NaradaBrokering: A Middleware Framework and Architecture for Enabling Durable Peer-to-Peer Grids. Proceedings of the ACM/IFIP/USENIX International Middleware Conference Middleware-2003. pp 41-61.

[2] The NaradaBrokering Project at the Community Grids Lab: http://www.naradabrokering.org

[3] Geoffrey Fox and Shrideep Pallickara. An Event Service to Support Grid Computational Environments Journal of concurrency and Computation: Practice & Experience. Volume 14(13-15) pp 1097-1129.

[4] A Simulation of the NaradaBrokering Topology Viewer. http://www.naradabrokering.org/tv/simulation/topology.htm

[5] I. Herman, G. Mclancon and M. S. Marshall, "Graph Visualization and Navigation in Information Visualization: a Survey", IEEE Transaction on Visualization and Computer Graphics, Vol. 6, pp. 24-43, 2000.

[6] G. di Battista, P. Eades, R. Tamassia, and I.G. Tollis, "Algorithms for drawing graphs: an annotated bibliography", Computational Geometry: Theory and Applications, Vol. 4, No. 5, pp. 235–282, 1994.

[7] Inas Khalifa, "Characterization of the Internet at the AS Level: Visualization and Modelling", ENSC 835 Project, 2002.

[8] Cooperative Association for Internet Data Analysis (CAIDA), MapNet and Otter Tools

[9] Nickie Buckner, "Visualization Techniques for Hierarchical Information Structures".

[10] B. Johnson and B. Schneiderman, "Tree-maps: a Space-filling Approach to the Visualization of Hierarchical Information Structures", Proceedings of IEEE Visualization '91, IEEE CS Press, pp. 275-282, 1991.

[11] Jason Baumgartner, Yuezheng Zou, and Katy Borner, "Space Filling or Treemap Algorithms", http://iv.slis.indiana.edu/treemap.html

[12] Ben Shneiderman & Martin Wattenberg, "Ordered Treemap Layouts", Proceedings of Infovis 2001, pp. 73-78, 2001.

[13] Robertson G.G., Mackinlay J.D, and Card S.K., "Cone Trees: Animated 3D Visualizations of Hierarchical Information", Proceedings of CHI '91. ACM, 1991. pp. 189-194.

[14] John Lamping and Ramana Rao, "Laying out and Visualizing Large Trees using a Hyperbolic Space", Proceedings of UIST '94, pp. 13-14, 1994.

[15] John Lamping, Ramana Rao, and Peter Pirolli, "A focus+context Technique based on Hyperbolic Geometry for Viewing Large Hierarchies", Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems, May 1995.

[16] T. Munzner, "H3: Laying out Large Directed Graphs in 3D Hyperbolic Space", Proceedings of the IEEE Symposium on Information Visualization (Infoviz '97), pp. 2-10, 1997.

[17] P. Eades and Q.-W. Feng, "Multilevel Visualization of Clustered Graphs", Proceedings of the Symposium on Graph Drawing GD '96, Springer-Verlag, pp. 101-112, 1997.

[18] K. Sugiyama and K. Misue, "Visualization of Structural Information: Automatic Drawing of Compound Diagraphs". IEEE Transactions on Systems, Man and Cybernetics, Vol. 21 No. 4, pp. 876-892, 1991.

[19] K. Misue and K. Sugiyama, "An Overview of Diagram Based Idea Organizer: D-Abductor. Technical Report IIAS-RR-93-3E, ISIS, Fujitsu Laboratories, 1993.

[20] S. C. North, "Drawing Ranked Diagraphs with Recursive Clusters", preprint, 1993. Software Systems and Research Center, AT&T Laboratories.

[21] Brendan Madden, Patrick Madden, Steve Powers, and Michael Himsolt, "Portable Graph Layout and Editing", Graph Drawing '95, volume 1027 of Lecture Notes in Computer Science, pp. 385-395. Springer-Verlag.

[22] W. Tutte, "How to Draw a Graph", Proceedings of the London Mathematical Society Vol. 3 No. 13, pp. 743-768, 1963.

[23] P. Eades and Q.-W. Feng, "Orthogonal Grid Drawing of Clustered Graphs", Technical Report 96-04, Department of Computer Science, The University of Newcastle, Australia, 1996.

[24] Tom Sawyer Software, "Graph Layout Toolkit", http://www.tomsawyer.com/glt/

[25] G. J. Wills, "NicheWorks – Interactive Visualization of Very Large Graphs", Proceeding of Graph Drawing '97, Springer-Verlag, pp. 403-414, 1997.

[26] Ka-Ping Yee, Danyel Fisher, Rachna Dhamija and Marti Hearst, "Animated Exploration of Dynamic Graphs with Radial Layout", Proceedings of the 2001 Symposium on IEEE Information Visualization, pp. 43-50, 2001.

[27] Soon Tee Teoh and Kwan-Liu Ma, "RINGS: A Technique for Visualizing Large Hierarchies", Proceedings of Graph Drawing '02, 2002.