Interacting with Transit-Stub Network Visualizations

James R. Eagan*, John Stasko†, and Ellen Zegura‡
GVU Center, College of Computing
Georgia Institute of Technology
Atlanta, GA 30332

Abstract

Real-world data networks are large, making them difficult to analyze. Thus, analysts often generate network models of a more tractable scale to perform simulations and analyses, but even these models need to be fairly large. Because these networks do not directly correspond to any particular network, it is often difficult for the user to construct a mental model of the network. We present a network model visualization system developed with networking researchers to help improve the design and analysis of these topologies. In particular, this system supports manipulation of the network layout based on hierarchical information; a novel display technique to reduce clutter around transit routers; and the mixture of manual and automatic interaction in the layout phase.

CR Categories: H.5.2 [Information Systems]: Information Interfaces and Presentation—User Interfaces

Keywords: network visualization, graph layout, graph manipulation

1 Introduction

Because of the scale of real-world networks, networking researchers typically use network models of a more manageable scale on which to perform analyses. Tools such as the Georgia Tech Internet Topology Modeler (GT-ITM) [Calvert et al. 1997] generate pseudo-random network topologies on which researchers can perform their analyses. These networks are pseudo-random in the sense that they are randomly generated within the constraints of various properties that have been identified as existing in many real-world networks. One limitation of these systems is that the output of the model generator is an abstract description of a network; the leading feature request for GT-ITM is "How can I *see* what this topology looks like?"

To aid in the analysis of these network models, we created the NetVizor system, a tool designed to visually display the network models generated by GT-ITM. In designing NetVizor, we met with networking researchers to identify the tasks and peculiarities of the particular problems they address when looking at network topologies. One problem in particular is the generation of a suitable layout for a network.

To help address the problem of graph layout, we propose a general method of attack to the layout problem that mixes automatic layout algorithms with manual interaction. Another problem that our networking participants face is the publication of generated models. As such, the aesthetics of the layout are important to convey the structure of the topology adequately. To help the user refine the layout, we take advantage of the hierarchical nature of real-world networks and use hierarchy information to aid in the manipulation of the layout of nodes and domains in the visualization. Lastly, we introduced a "fudge-factor" in the visualization that adds virtual aggregate edges to the visualization to reduce clutter around transit domains. We discuss these three techniques in more detail in the next few sections.

*email: eaganj@cc.gatech.edu
†email: stasko@cc.gatech.edu
‡email: ewz@cc.gatech.edu

2 Related Work

Although the network topologies we are working with are not general graphs, work in the field of graph layout is relevant. A lot of work has gone into this field [Battista et al. 1999]. We leverage this existing work, focusing instead on the application of these techniques to this particular network layout problem.

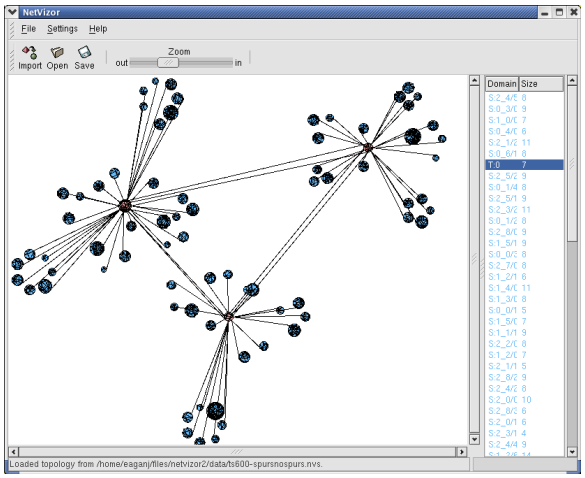
The Nicheworks system [Wills 1999] and the H3 browser [Munzner 1997]operate on arbitrary graphs, but do not provide explicit support for hierarchical or nested graphs like the ones generated by GT-ITM. The layouts generated by Nicheworks are primarily static with respect to manual repositioning of the nodes within the graph. The H3 browser supports good interaction with the graph, but the layout is fixed in its hyperbolic space — the user changes perspective on the graph rather than how everything is laid out.

The GraphVisualizer3D (GV3D) system [Ware et al. 1997] and the HINTS system [do Nascimento and Eades 2001] each involve the user in the layout process. In GV3D, the user plays a cleanup role in the layout process, *post hoc*. In the HINTS system, the user provides hints about the structure of the graph to improve the performance of the automatic layout algorithm for the purposes of generating a better layout. No emphasis is placed on improving the user's understanding of the structure of the topology.

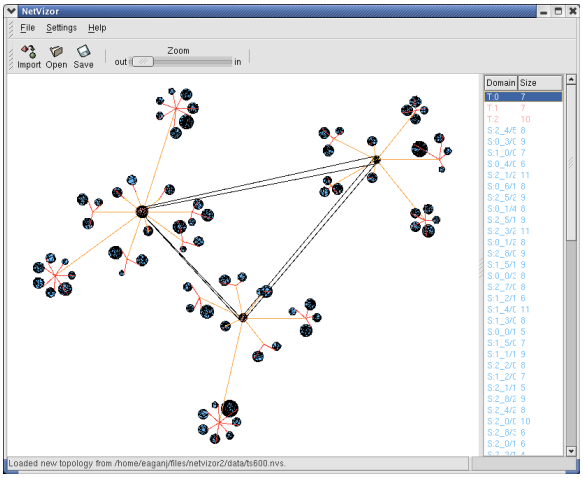
Nam [Estrin et al. 1999], the network animator, provides an animation of a network animation trace, but has very rudimentary layout and interaction capabilities; its focus lies on the animation of trace data. Tools such as the Extended Nam Editor [nam 2003] provide more robust editing capabilities.

3 Transit-Stub Models

The models generated by GT-ITM follow the transit-stub model of networks. In this model, nodes, which represent routers on the network, are organized into logical domains, or collections of nodes. Nodes within a domain tend to be fairly interconnected within the domain, but rarely connect to nodes outside of the domain. Domains themselves are then classified into two types: transit domains and stub domains. Nodes in a stub domain are typically an endpoint in a network flow — network traffic either originates at or is destined for a node in a stub domain. Nodes in transit domains are typically intermediate in a network flow — traffic is typically just passing through. For example, one of UUnet's backbone routers would be in a transit domain, while a router at the local ISP would be in a stub domain.



(a) Traditional Graph View



(b) Spurred Graph View

4 Manual-Automatic Hybrid Layout

We suspect that mixing manual interaction with automatic layout can help the user of the system forge a stronger mental understanding of the structure of the model topology. This aid is particularly important in this case because the topologies being presented do not directly correspond to any existing real-world network. By letting the user do some of the work, he or she can better understand the process that is taking place and the structure of the network; by doing most of the work automatically, the system can keep the task from becoming too tedious. Thus, the user can "sketch out" a high-level overview of the layout, while the system fills in the details.

When loading a new topology, the system presents the user with 3 options: layout the network automatically; layout the network manually; or layout the network using a mixture of the two. In the last case, the user is presented with a blank canvas and a list of the domains in the network. The user then assigns a position to each transit domain in the network; as a position is defined, the system runs an automatic layout algorithm on the stub domains that peer with that transit domain and on all of the nodes within each of those domains. In the case of a 2000 node topology, the manual component of the layout process consists of laying out 10-15 transit domains in a typical case.

5 Aggregation Spurs

Typically, many stub domains connect to a single transit node in a transit domain, with many other stub domains connecting to the other nodes within the transit domain. When drawn on the screen, this creates a ball of string as many edges converge on a small location on the screen. To help combat this problem, we introduce a virtual aggregation edge, which we call a "spur" to the network. Each spur draws a transit node outside of the domain and creates a larger area for all of the stub peers of a transit domain to converge upon (See figure 1).

6 Hierarchical Manipulation

We take advantage of the hierarchical nature of the transit-stub model when manipulating the layout of the graph. When the user drags a node on the screen, its position is constrained within the domain it is in. When a domain is moved on the screen, all of the nodes within the domain move with it, as the user would expect. When the user changes the position of a transit domain, however, all of the stub domains that peer with it move as well, in addition to the nodes within the domains. Thus, one reposition of the transit domain can move the entire group of domains associated with that domain, as the user would typically wish to do. Similarly, when the user adjusts the position of one of the spurs, all of the domains that peer with that node are repositioned.

References

BATTISTA, G. D., EADES, P., TAMASSIA, R., AND TOLLIS, I. G. 1999. *Graph Drawing — Algorithms for the Visualization of Graphs*. Prentice Hall.

CALVERT, K., DOAR, M., AND ZEGURA, E. W. 1997. Modeling internet topology. *IEEE Communications Magazine* (June).

DO NASCIMENTO, H. A. D., AND EADES, P. 2001. A system for graph clustering based on user hints. In *Pan-Sydney Workshop on Visual Information Processing*.

ESTRIN, D., HANDLEY, M., HEIDEMANN, J., MCCANNE, S., XU, Y., AND YU, H. 1999. Network visualization with the vint network animator nam. Tech. Rep. 99-703, University of Southern California.

MUNZNER, T. 1997. H3: Laying out large directed graphs in 3d hyperbolic space. In *IEEE Symposium on Information Visualization*, 2–10.

2003. Extended Nam Editor.

WARE, C., FRANCK, G., PARKHI, M., AND DUDLEY, T. 1997. Layout for visualizing large software structures in 3d. In *Visual97 Second International Conference on Visual Information Systems*, 215–225.

WILLS, G. J. 1999. Nicheworks — interactive visualization of very large graphs. *Journal of Computational and Graphical Statistics 8*, 2, 190–212.