First publ. in: Proc. AT&T Software Symposium, 1999

Visualizing Large-Scale Telecommunication Networks and Services

James Abello
Emden R. Gansner
Daniel Keim
Eleftherios Koutsofios
Stephen C. North
Russ Truscott
Network Services Research Lab
{abello,erg,keim,ek,north,truscott}@research.att.com
http://infolab.research.att.com

1 Introduction

Global telecommunication networks and services are among the enterprises having the highest volumes of real-time data. A voice network may complete more than 250 million calls per day. Each is described by one or more events, yielding a total of tens of gigabytes of data daily. Wireless, Asynchronous Transfer Mode (ATM), frame relay, Internet Protocol (IP) networks and higher-level services on them also are described by massive data sets, and can present additional problems in reconstructing an end-to-end view of user activity. Understanding this data at full scale is crucial for managing networks and improving their performance and reliability from a customer's viewpoint. Visualization techniques have become increasingly important to achieving this goal.

In the AT&T Infolab, we have developed a new visualization platform for the interactive data exploration of large networks. The platform incorporates interactive 3D maps and diagrams, statistical displays, network topology diagrams, and pixel-oriented displays, while supporting a variety of display technology, including a wall-sized screen.

Its applications include monitoring and analyzing activities at the network element, network-wide, customer and service levels. These activities may be network generated (e.g., exploration of network events and alarms or customer generated (e.g., usage anomalies such as fraud). End uses of the analysis would likely include improvement of service to customers, market analysis and gaining an understanding of previously hidden relationships between and within data segments.

In this paper, after an initial description of the types of applications the platform is aimed at, we concentrate on three novel aspects of the system. We describe our use of large displays as one approach to handling large data sets. We then describe the

SWIFT-3D network viewer, which provides the main software base for the platform. Finally, we consider a new visual metaphor for viewing large networks.

2 Telecommunication Applications

The goal of this work is to support interactive visual exploration of databases that describe full-scale commercial telecommunication networks, and to simultaneously raise the level of abstraction in visualization, for example showing layered services or network performance from an individual customer's viewpoint. Derived from this goal is the ability to move from data to business decision within minutes.

Many data analysis tasks that are tractable on small or medium-sized data sets can be difficult at greater scale. When practitioners refer to terabyte databases, they sometimes mean databases of image, sound or video data. In contrast, our application involves working with many small records describing transactions and network status events. The data processing involved is different in terms of the number of records and data items to be interpreted. In voice networks, the detail record for each call conforms to an industry standard format (Automatic Message Accounting, or AMA) that has about 50 attributes such as originating and terminating phone numbers, date, time and duration of the call. In our application this information is stored for each of the hundreds of millions of calls made daily, yielding about 15 GByte of data uncompressed. In addition, data is collected from the other networks previously mentioned. Understanding the relationships between them is increasingly important, e.g. to manage integrated communication services for global enterprises, but the data management problems that result are even more challenging than for a single service.

More than just scale is involved: our goal is also to raise the level of abstraction in network visualization, and to improve the real-time response of our analyses. This can help network managers and business decision makers to recognize and respond to changing conditions quickly; within minutes when possible. A scalable research prototype for visually exploring full-scale network traffic must therefore provide good interactive response, avoid instance-specific processing, and be flexible enough to support experiments in both back-end queries and the user interface. In our initial experiments, we found that commercial database systems either couldn't handle such large volumes or consumed far too many resources. Another problem with commercial databases is that the administrative effort was too high to support experimental research. Databases, however, have many useful features, such as data independence, and a standard query language. The tools we have built have some of these features. A main difference in our approach is the emphasis on data streaming, in comparison to a query/response methodology employed by formal databases.

3 Large Displays

One approach to the scale problem is to use a physically large display. Figure 1 shows our display wall, inspired by projects such as the CAVE[7] and Powerwall[8].

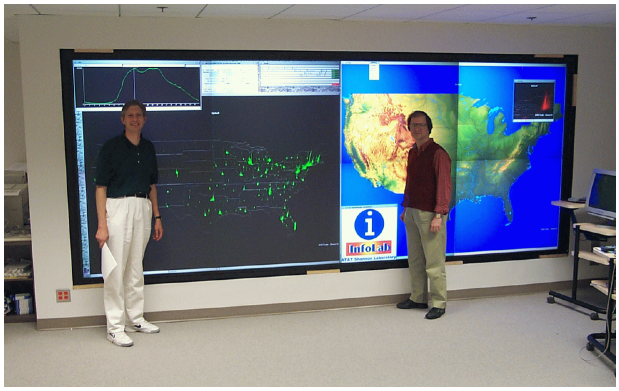

Figure 1: The Infolab Wall

Our main display wall ($6' \times 15'$) is driven by 8 LCD projectors. These are connected through a software-controlled video switch, usually to display the output of two graphics pipes of an SGI Onyx. The same Onyx can drive a smaller ($7' \times 9'$) 4-projector wall elsewhere in the same building, using its third pipe. Other compute and disk servers for network data analysis projects are connected on an 800-megabit

High Performance Parallel Interface (HIPPI) network, providing 10 terabytes of on-line storage and another 20 terabytes of tape under hierarchical storage management.

4 Visualization in SWIFT-3D

The principal application that runs on the display wall is SWIFT-3D, an interactive, large-scale network viewer [10]. The SWIFT-3D system integrates a collection of relevant visualization techniques ranging from familiar statistical displays, to pixel-oriented overviews with interactive 3D-maps and drag+drop query tools. It provides comprehensive support for data exploration, integrating large scale data visualization with querying, browsing and statistical evaluation (see [3, 4, 5, 11] for examples of previous related work). The visualization component maps the data to a set of linked 2D and 3D views [6, 12] created by different visualization techniques:

- Statistical 2D Visualizations (line graphs, histograms, etc.) - used as overview displays and for interactive data selection
- Pixel-oriented 2D Visualizations - intended as bird's-eye overviews and for navigation in 3D displays
- Dynamic 3D Visualizations - used for an interactive detailed viewing of the data from different perspectives.

In addition, the system provides tightly integrated browsing and querying tools to select the data to be displayed and to drill-down for details if some interesting pattern has been found.

A screenshot of the overall system is given in Figure 2. The upper left window shows a time line visualization of voice network volume in 10-minute intervals. This plot shows the volume for different services (e.g. residential, business, and 1+ dial-around service, software-defined networks, and aggregate volume). The window below the time line allows the user to select data for display by date, time or type of service. The large window shows a three-dimensional display of the data using a histogram spike for each location to display a value (typically, level of activity) corresponding to the cursor position in the time line window (11:00). The user can interactively navigate in the 3D display, zoom in at interesting locations, or view the map from arbitrary perspectives. An automated path-planning module has also been designed to determine a natural, context-preserving path from one viewpoint to another. The mapping between the data and display objects is set in an auxiliary file that contains geometric information about points, lines, polygons, and triangles, and coloring. Various color maps may be defined to highlight interesting properties of data. The mapping file may contain multiple levels of detail; for example, a data set representing the United States may be divided according to state, county, and telephone exchange, census block and 9-digit postal zip code outlines. Also, multiple data value sets can be mapped to the same geometry. For example, we can map state population to the state outline level and county population to the county level. As the view of a state enlarges, the displays can shift from showing a single value for state population to showing one per county. The user may also play through an adjustable interval in the time line window to get an animated time-sequence display (see video). If the user sees an interesting pattern in the visualization window, a drag-and-drop interface is available to drill-down to get details, explore context and take actions if necessary. This provides an intuitive way of converting spatial information into detailed information such as the top originating or top dialed numbers.

An additional 2D overview window is provided, showing call volume for each location by one colored pixel (cf. lower left corner of Figure 2). The technique behind the pixel-oriented 2D overviews is an adaptation of the Gridfit approach used in the VisualPoints system [9]. Gridfit places data points on a pixelated display, so that points having coordinates that would normally map to the same display pixel are represented by other pixels that would otherwise be unoccupied. Its algorithm is based on hierarchical partitioning of the data space, using a top-down reallocation of the screen space according to the requirements of subregions. Gridfit allows an efficient

and effective repositioning of the pixels on the screen such that the (absolute and relative) position of the data points and their distance is preserved as much as possible. The color is chosen such that high call volumes are mapped to dark colors and low call volumes are mapped to bright colors.

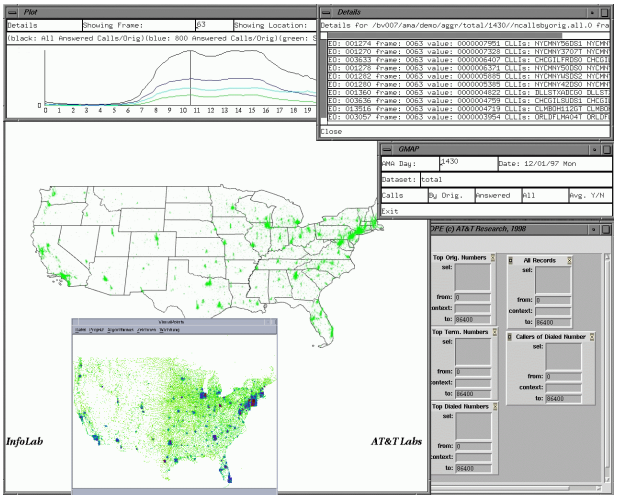

Figure 2: SWIFT Overview Screen

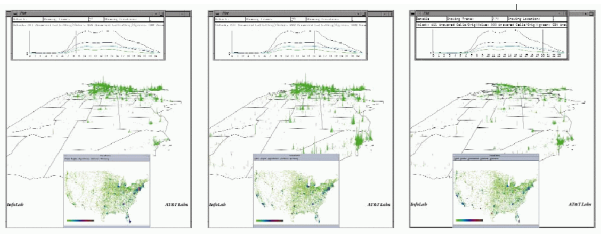

Figure 3: SWIFT Time Series

4.1 Implementation Architecture

SWIFT-3D consists of three modules: data collector, aggregator, and visualization interface. These communicate using self-describing data-independent binary formats consisting of a header that defines record size, type, and data context, followed by the actual data. This is necessary since SWIFT-3D is designed to work in real-time: the data processing modules can work incrementally and the visualization tools can safely access the data files while they are being updated.

To achieve good performance, SWIFT-3D uses various techniques that minimize processing delays and the use of system resources. Such techniques include processing pipelines, direct IO, memory mapping, and dynamic linking of on-the-fly generated code.

4.1.1 Data Collection and Storage

SWIFT-3D must collect data from many different sources having their own specialized formats. SWIFT-3D includes tools to convert such data to its internal self-describing format. When data is already in a fixed format, all that needed is to associate a data record schema with the file. In most cases, standard tools suffice for data conversion, but some types of data need more intricate pre-processing. For example, the voice network generates records in the previously-mentioned AMA format. This format has many sub-record types that can be combined to describe a call. Extracting information from AMA files is further complicated because, depending on the type of call, a value can be stored in different sub-records. For example, the dialed number is kept in different places in domestic and international calls. Such idiosyncratic processing is performed by custom tools to load into SWIFT-3D format.

4.1.2 Data Processing

Initial processing of a data feed usually involves reading in records and computing basic statistics. SWIFT-3D relies on a stream pipeline model. Accessing large-scale data on disk can be expensive, so instead of storing the output of each processing step to disk, the stream processors are implemented as concurrent processes that exchange data. SWIFT-3D extends the UNIX pipe model of single writer and single reader to that of single writer and many readers to minimize data copying. For example, we might need to process a day's worth of telephony data and compute: (1) how many calls there were per area code and exchange (NPA/NXX), (2) how many calls did not complete and separate these by failure type. This could be implemented by a single process that

reads data from disk and feeds two other processes to count.

SWIFT-3D provides several tools to be used in such pipelines. These include tools to filter records (e.g. remove calls that did not complete), count based on attributes (e.g. count number of incomplete calls by NPA/NXX), split a single file into several based again on some combination of attributes (e.g. separate calls into a file per type of service such as toll free calls, operator calls, collect calls, etc.).

The expressions used for filtering, counting, and splitting are specified as C-style expressions. For example, the expression

```
if (tos == TOLLFREE && iscomplete) KEEP;
else DROP;
```

filters out calls that are not toll free (1-800, 1-888, 1-877) or not complete. These expressions are turned into C code that is compiled into shared objects on the fly and are then dynamically linked in and executed. This approach combines the speed of compiled code with the flexibility of tools such as AWK. C and AWK seem to be the tools of choice for most of our statisticians and analysts.

4.1.3 Data Visualization

SWIFT-3D's visualization tools allow users to explore data filtered by the stream processors. They are designed to be interactive in the sense that the user can view some data set, focus on a specific subset, query the system for the raw data that generated this subset, re-aggregate and view the new result. SWIFT-3D enables this by keeping enough information to link raw data, aggregate data and visual objects. This link to visual objects is implemented by generating geometric data sets that contain information about the items they represent. For example, an NPA/NXX may be represented by a point, bar, or polygon of its geographic area. In all cases, the geometry file contains information to link the NPA/NXX to the point, line, or polygon. Besides answering user queries, this facility is also used to alter the geometry based on data values. For example, if NPA/NXXs are shown as polygons and busy NPA/NXXs need to be colored red, the system uses this mapping to determine red polygons.

For reading records off disks, SWIFT-3D uses Direct-IO if available. Direct-IO bypasses kernel buffer copying from disk, and can be twice as fast as normal IO. (Normal IO can be faster for data that was recently read and is still in cache, but given the size of our datasets, this is rarely the case).

The format of the counts files is also self-describing. Such files implement a type of two dimensional array of values (integers, floats, etc.). One dimension (the 'frame') corresponds to time buckets while the other (the 'item') corresponds to the aggregation type. The second dimension can be accessed using a dictionary that maps item ids to item positions. For example, the second dimension in the scenario above may have an entry per NPA/NXX observed, and the dictionary might indicate that data for NPA/NXX 973-360 is in position #0.

These files are designed to be accessed and changed incrementally: when new data arrives, these files are opened and the various counts are increased in place (using some buffering to minimize accesses to the files). The actual updating of the files is done using memory mapping, due to the random access nature of the updating. File locking is used to protect against accessing such a file in the middle of an update. Also, each update increments a count stored in the file. This allows the visualization tools to efficiently check if the file has been modified.

4.2 SWIFT-3D Applications

SWIFT-3D has been applied to several different problems in network visualization. These include the ability to provide an abstraction that permits visualization of the data across the information strata of network element, network, services and customers; the ability to view cross network interactions and their impact upon a service and/or customer; the capabilities to discern impact on one or more customers when there is a network event.

An interesting example is the examination of calls that cannot be completed due to congestion at the

customer premise. Keeping this number low is important due to the resources consumed. It is important both to the customers (who need reliable service for telemarketing sales and customer support) and to AT&T from a financial standpoint (because unanswered calls consume network resources and may incur cross-carrier settlement charges without creating revenue). In visually exploring voice network events, we noticed that on several days within an interval of several weeks, many unanswered calls originated in a certain metropolitan area (cf. Figure 4). The events always occurred at bottom of the hour (:30) for several hours in the evening. By interactive querying we found that most of the calls were directed at one 800 number, and that the number belonged to a radio station. By tuning in, we discovered that the station was giving out free tickets for an upcoming concert. The winner was the tenth caller at the bottom of each hour.

Another application concerns analysis of an Internet service. There is considerable motivation for understanding relationships between usage of multiple services, both from a single service provider, and between competitors. AT&T wanted to know how much coverage an Internet access service had. The coverage is measured by the number of the area code and exchanges (and ultimately households or customers) where connecting to the Internet is a local call (usually without per-minute charges). We contracted two companies that claimed to have such data. We gave them the locations of the modem pools and asked them to tell us what codes and exchanges were covered. We received two very different answers. To understand the differences we used SWIFT-3D: areas claimed to be covered in the answer of company A were colored blue, those claimed to be covered in the answer of Company B were colored green, where both companies agreed, the map was colored gray. We noticed widespread differences in many states, while a few states had good matches. In order to decide which company's answer was more correct, we superimposed our customer usage data on the map. In the generated visualizations (cf. Figure 5), we saw that there was a lot of usage in gray and blue areas, but very little usage in green (and almost none in black areas). Our conclusion was that the answer by company A was more correct. It further became clear that individual customers are very aware of local calling areas, and are not willing to use an ISP when the access would be too expensive. A side effect of our findings was that the business decided to not even advertise this service in areas not covered by local access.

A third application involves recognizing the characteristics of virtual private networks (VPNs) provisioned by customers on a large packet network, and their relationships to physical network facilities. Figure 6 shows the peak volume of Permanent Virtual Circuit (PVC) traffic, by VPN, for the whole network in one 5-minute period. The display highlights the PVCs having the greatest load. The eventual goal of this study is to understand customer-focused, near real-time management of packet networks.

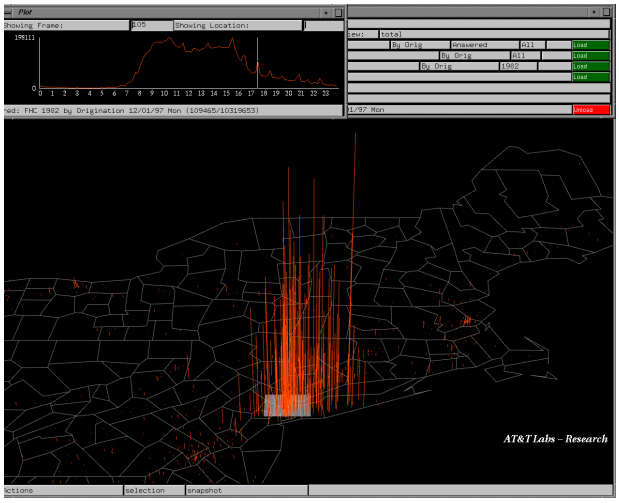

Figure 4: Network event analysis

It should be noted that, even with the disparity among these application domains, it was not difficult to tailor SWIFT-3D to each. In large measure, this is due to the similarity of the visual models, and to the high-level descriptions used to specify much of the analysis and display. Probably the most difficult aspect in modifying SWIFT-3D for an application is construction of tools to massage the application's data into a format suitable for SWIFT.

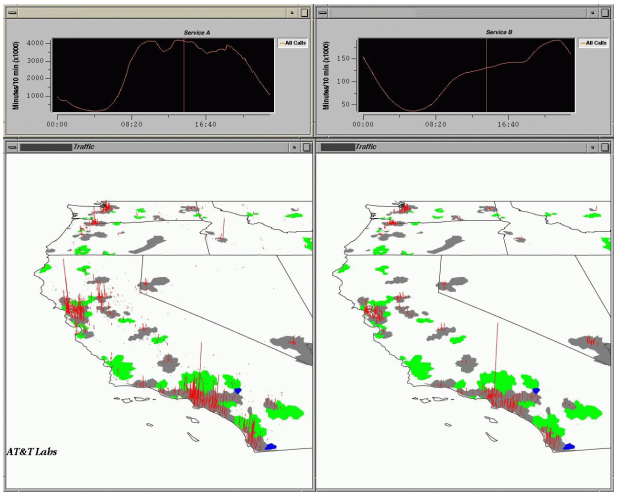

Figure 5: Geographic coverage analysis

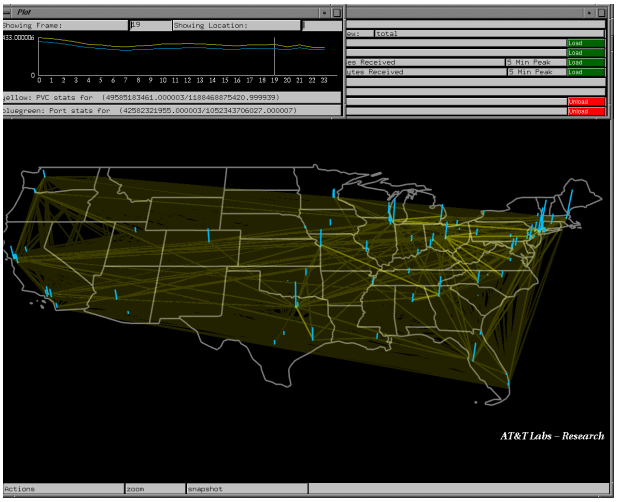

Figure 6: Packet traffic analysis

5 Graph Surfaces

As the size of the data sets increases, the optimizations within SWIFT-3D and the use of large displays become inadequate. What is needed is a new way to visualize aggregated data. One approach is to use graph surfaces[1], which provide a new visual metaphor for large networks. They provide a collection of natural operations for browsing at different levels of detail, and fit well in the context of large networks, allowing convenient incremental update and navigation of external memory graphs (cf. [2]) whose vertex sets are hierarchically labeled, as is typical with call detail and Internet data.

The main idea is to view a network as a discretization of a 2D surface in 3-space. For a fixed ordering of the network components, a rectangular domain is triangulated and each point is lifted to a height corresponding to aggregated attribute of the underlying network. This provides a piecewise linear continuous function forming a polyhedral terrain. The terrain is used as an approximation to a surface representing the communication network. An example is shown in Figure 7.

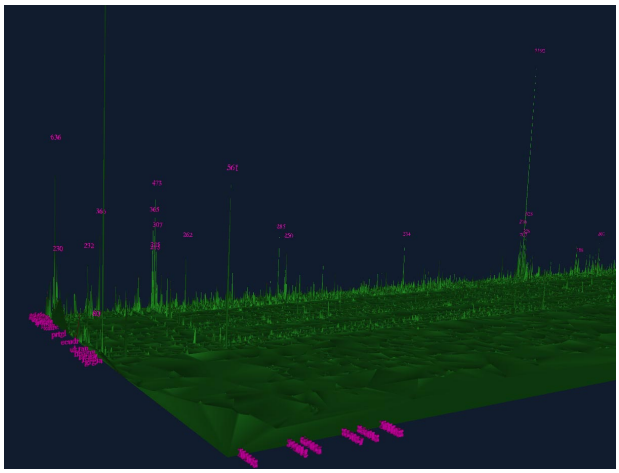

Figure 7: A graph surface

In order to handle very large communication networks, a hierarchy of surfaces is constructed. Each represents an aggregate view of the network at a lower level. Operations are provided that allow the user to navigate the surfaces.

5.1 Constructing and navigating a graph surface

A network, during some interval of time, can be viewed as a collection of vertices, a collection of edges connecting the vertices, and weights on the edges, measuring some attribute of an edge. For example, in the case of call detail data, the vertices correspond to phone numbers.

An important aspect of these networks is that the vertices can be viewed as having an underlying hierarchical structure. This induces a hierarchical structure on the edges. At the lowest level, one has the original network. Subsequent levels are obtained by coalescing disjoint sets of vertices at a previous level and aggregating their corresponding weighted edges. Each level in the hierarchy can be represented as a surface and the hierarchy can be used as a road map to move from surface to surface. By collapsing surfaces into edges and expanding edges into surfaces, one gets zoom in and zoom out operations, as exhibited in Figure 8. An important point, for large networks, is that the hierarchy can be constructed efficiently. (See [1] for details.)

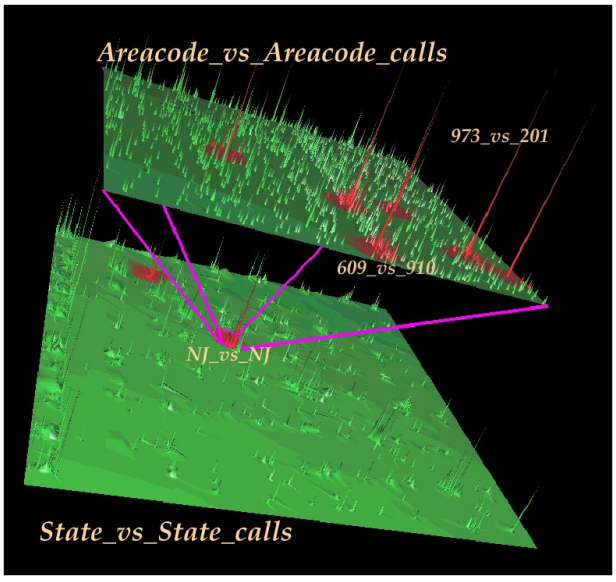

Figure 8: Graph surface navigation

5.2 Implementation

To handle large networks effectively, there are three cases to consider:

- The hierarchy fits in main memory
- The hierarchy does not fit but the vertex set does
- The vertex set does not fit

In the first case, the edges are read in blocks and each one is filtered up, through the levels of the hierarchy, until it lands in its final level. This can be achieved with one pass over the data. The second and third case are handled by setting up a multilevel external memory index to represent the hierarchy. Filtering the edges requires now several passes over the data. The I/O performance depends strictly on the I/O efficiency of the access structure.

The visualization of a data set that is several orders of magnitude more than the available screen resolution calls for some form of hierarchical graph decomposition. Graph surfaces by definition provide a geometric view of a very large graph in a manner that is amenable to hierarchical browsing via the zoom in and zoom out operations. This makes it feasible to explore deeper hierarchy elements at a finer level of detail. Also, a graph surface provides both a uniform local and global view.

The graphical engine generates polyhedral terrains that correspond to individual edges in hierarchy. Initially, a suitable level in the hierarchy is chosen as the root, depending of the available screen resolution. The polyhedral terrains are generated with a triangulation algorithm and are displayed using several visual cues and dynamically generated labels. The graphical engine is implemented in C++ and uses the OpenGL standard library for the rendering portion. Currently, a mouse/keyboard interface is used. The use of joysticks and gestures to navigate the environment is currently under consideration.

5.3 Applications

Currently, graph surfaces are being used experimentally for the analysis of several large multi-digraphs

arising from the AT&T network. These graphs are collected incrementally. For example, in the call-detail multi-graph, vertices correspond to phone numbers and edges represent a call from one number to another. The graph grows by daily increments of about 300 million edges, defined on a vertex set on the order of 300 million vertices. The aim is to process and visualize this type of multi-digraph at a rate of at least a million "edges" per second.

Internet data is another prime example of a hierarchically labeled multi-digraph that fits the graph surfaces metaphor quite naturally. Each i-slice represents traffic among the aggregate elements that lie at the i^{th} level of the hierarchy. The navigation operations can be enhanced to perform a variety of statistical computations in an incremental manner. These in turn can be used to animate the traffic behavior through time.

When the vertices of the multi-digraph have an underlying geographical location, they can be mapped into a linear order using, for example, Peano-Hilbert curves, in order to maintain some degree of geographical proximity. In this way, the constructed surface maintains a certain degree of correlation with the underlying geography.

6 Conclusions

The Infolab work, with its heterogeneous displays, its flexible and powerful software base, and its new visual metaphors, has proved successful in giving its users a new way of viewing AT&T networks, and serves as a starting point for much future work.

At the simplest level, the system needs to be packaged so that it can be used on a daily basis by network analysts and administrators. The system should be configured so that a user can easily modify it for specialized display and analysis needs. This work is already underway for applications in NOC-2000 and the AT&T frame relay network. In addition, the system needs to be available on the desktop. To address this last point, we have a prototype implementation in Java and Java-3d, where the user interfaces runs on a PC and more powerful servers handle data processing.

Another avenue of work is to enhance the system to better handle the heterogeneity of AT&T's networks and workplace. It should be possible for employees, at different locations with different display environments, to visually share network information. At the same time, the many types of networks could be logically integrated into a single network.

Our display wall has proven itself. However, its current set of input devices (mouse and keyboard) seems inadequate. We would like to explore new sources of input, such as wands, gesture recognition and palm computers.

Many of the views in the system are, at present, based on the underlying geography. This is a significant limitation for network visualization. Often, many endpoints are located in a few dense metropolitan areas, with large 'deserts' in between, so maps do not use the available pixels efficiently. Also, maps work well for displays of endpoints (vertices) but not so well for edges (vertex pairs), let alone groups of more complex structures such as routes, flows, or subgraphs representing virtual networks. With IP networks, geographic coordinates are not always even available. One can move to more abstract topological representations, but the standard visualization techniques rapidly become unusable as the data grows, even though a megapixel display delays this somewhat. There is need for additional work in display metaphors and interaction techniques for large graphs. Graph surfaces, described above, area one approach, but additional new ways of "looking" at networks and network data are called for.

7 Acknowledgements

For collaboration on experiments and support of technology transfer, the authors thank Ed Berberich, Chee Ching and John Ennis from NOC-2000, and Arvind Chakravarti, Ram Chelluri, Quynh Nguyen and John Medamana from the AT&T Frame Relay Network lab.

References

[1] J. Abello and S. Krishnan. Graph Surfaces. In *Int. Conf. Industrial and Applied Math.*, July 1999.

[2] J. Abello and J. Vitter, editors. *External Memory Algorithms*. AMS-DIMACS, 1999.

[3] C. Ahlberg and E. Wistrand. IVEE: An Environment for Automatic Creation of Dynamic Queries Applications. In *Proc. ACM CHI Conf. on Human Factors in Computing (CHI95)*, 1995.

[4] C. Ahlberg and E. Wistrand. IVEE: An Information Visualization and Exploration Environment. In *Proc. IEEE Int. Symp. on Information Visualization*, pages 66–73, 1995.

[5] Richard A. Becker, Stephen G. Eick, and Allan R. Wilks. Visualizing network data. *IEEE Transactions on Visualization and Computer Graphics*, 1(1):16–28, March 1995.

[6] A. Buja, J.A. McDonald, J. Michalak, and W. Stuetzle. Interactive Data Visualization Using Focusing and Linking. In *Proc. IEEE Visualization*, pages 156–163, 1991.

[7] C. Cruz-Neira, D. Sandin, and T. DeFanti. Surround-Screen Projection-Based Virtual Reality: The Design and Implementation of the CAVE. In *Siggraph 1993 Conference Proceedings*, pages 35–142, 1993.

[8] Paul Woodward et al. University of Minnesota PowerWall, 1998. http://www.lcse.umn.edu-/research/powerwall/powerwall.html.

[9] Daniel A. Keim and Annemarie Herrmann. The Gridfit Algorithm: An Efficient and Effective Algorithm for Visualizing Large Amounts of Spatial Data. In *Proc. IEEE Visualization*, pages 181–188, 1998.

[10] Eleftherios E. Koutsofios, Stephen C. North, and Daniel A. Keim. Visual exploration of large telecommunication data sets. *IEEE Computer Graphics and Applications*, 19(3):16–19, May 1999.

[11] S. F. Roth, P. Lucas, and C. C. Gomberg. Visage: Dynamic Information Exploration. In *Proc. ACM CHI Conf. on Human Factors in Computing (CHI96)*, April 1996.

[12] M. O. Ward. XmdvTool: Integrating Multiple Methods for Visualizing Multivariate Data. In *Proc. IEEE Visualization*, pages 326–336, 1994.