Multi-View Visualization of Simulated Network Data

Derek Overby¹, John Keyser², Jim Wall¹

¹Computing and Information Technology Division
Texas Center for Applied Technology
Texas A&M University System

²Department of Computer Science
Texas A&M University

Abstract
Researchers and professionals in the military community are challenged to provide useful information to decision-makers from massive collections of data which are the result of large scale simulation exercises. Recently some System-of-Systems (SoS) level simulation exercises have produced datasets in excess of 20 terabytes. As these Systems-of-Systems increasingly become more complex, and as we continue to simulate and test the interoperability of these systems in order to support the development and operational testing of net-centric components of the Future Combat System (FCS), we require methods to enable analysts to discover meaningful information within very large datasets. We propose an application architecture in which multiple visualization capabilities can be customized in an interactive environment in order to promote data exploration and visual analysis. Flexible visualization capabilities will improve the knowledge discovery process by allowing the user to adjust properties of processing and visualization methods without developer interaction. The software framework we have developed provides extensible data visualization capabilities via a plug-in architecture and allows the user to employ multiple visualization techniques to enhance the information value of a dataset by customizing each visualization to provide information which complements other visualization methods. In addition, by linking user interactions to multiple visualization methods via common controls, we allow the analyst to explore different perspectives of the same data synchronously.

1 Introduction
Analysis of network performance is critical to the development, acquisition, and validation of future military systems, many of which are designed to operate in net-centric environments. While many commercial software solutions do provide network visualization capabilities, these applications are often limited in terms of user interaction and do not provide the degree of customization needed for analysts to conduct visual data mining and knowledge discovery after the event. In addition, changing requirements and test environments lead to new data configurations that may often require costly developer modifications to these applications.

Contemporary researchers and professionals have argued that providing applications which allow the analyst to engage in ‘discourse’ with data is necessary to allow human analysts to obtain critical information from massive volumes of data in a timely manner [1]. As recent history demonstrates, the availability of accurate and timely information can be critical to decision makers in times of crisis. Within the development and acquisition process, timely analysis and anomaly identification in network datasets can save both the military and defense contractors large amounts of time and money by quickly identifying faulty or misconfigured network equipment during large-scale testing events.

We propose that providing multiple visualization capabilities within an extensible application framework can enhance the analyst’s ability to detect the expected and discover the unexpected in these very large datasets. The software framework we have developed allows the user to configure custom data visualizations by interactively connecting a set of capabilities provided by the framework. Data access and transformation components can be easily reconfigured by end users to adapt existing capabilities to be used on new datasets. Additionally, user interactions are correlated with all visualizations and therefore users can use multiple visualization methods to analyze different aspects of data synchronously. This correlation can increase the amount of information available to the analyst at any one time by providing him or her with different perspectives of the same data in a linked fashion.
2 Framework Overview

The software framework we have developed is based on an abstraction of the data processing and visualization process, shown in Figure 1. The generalized steps of the process are: access data, transform data, process data, visualize and interact with data. The primary components of the framework are shown along the top of the figure, corresponding to these generalized steps. Various implementations of these components are added to the application framework by developers to achieve specific visualization methods, as shown at the bottom of the figure. A more detailed description of the framework architecture and implementation is available here [2].

Compared with previously proposed frameworks for visualization systems, our framework differs in several ways. First, our framework focuses on defining an interactive architecture for visualization systems that end-users (i.e., data analysts) can use. Therefore we strive for simplicity in the overall framework and classification of objects. Second, our framework incorporates data access, transformation, processing, and visualization capabilities. Other frameworks provide only a subset of these function categories, most assuming either that data is transformed or processed beforehand. And finally, our framework defines components at a high-level only, therefore the low-level implementation of each component is left up to the developer, who can choose to employ formalized object-oriented programming concepts or rapid development for experimental methods.

All components within our framework can be classified as either a data contributor, a data consumer, or both. The only exception to this rule is the Adapter component, which will be discussed later. Shown in Figure 2, Contributors are components which can contribute data records to other objects in the framework while Consumers are components which can receive data from other objects. Contributors can define any number of outputs, and each output can define multiple directed connections to other objects. Consumers, on the other hand, can define any number of inputs however each input can only receive one directed connection from another object. Additionally each component must define a specific data type (e.g., string, integer) for every input and output.

2.1 Importers and Data Sources

An importer is a component within our framework which is responsible for providing access to a specific type of data source. An importer acts as a contributor only, and must be configured with connection information before it can be used. We refer to each configured instance of an importer as a Data Source. When importing data from a Relational Database Management System (RDBMS), one data source instance will be created for each table imported from the database.

Examples of importers include those for various implementations of network-accessible SQL databases as well as local databases such as Microsoft Access. Most importers need only to wrap over an existing database driver which handles any proprietary interface protocols necessary to connect and access the data. However importers for raw data such
as data in a text file must implement custom parsing methods to generate valid data records from the data source.

2.2 Operators and Adapters
Data operators are components in the framework which act as both consumers and contributors. A general operator, shown in Figure 3, can define any number of typed inputs and outputs and exhibit the previously described connection limitations of contributors and consumers. Operators can range from implementations of very simple mathematic or logic operations to complex operations that require internal cached data structures.

![Figure 3: Data Operator](image)

An adapter is a data mutator that operates on an input to an operator. Adapters are used to cast a data value from one type to another (e.g., string to integer), to match an operator's defined input type. Adapters are unique in because they do not behave like standard contributors in that their output can only be mapped to one input. Essentially, adapters are used to manipulate fields of data while operators are used to manipulate records of data.

2.3 Processing Module
The Processing Module is a framework component that is responsible for processing data from the incoming stream into one or more intermediate data structures. The processing module defines a set of input parameters and types. For each record in the data stream, a computation is performed on the input data and then the intermediate data cache is updated. Processing modules can be implemented to perform a variety of tasks ranging from simple numeric computations to complex statistical analysis. Multiple processing modules can also be used to perform more complex tasks, and therefore processing modules should be developed using the principles of abstraction and object-oriented design to facilitate reuse and decrease development effort in the future.

Additionally the framework provides a caching service that allows processing modules to cache and reload intermediate data products to reduce future processing. Changes in the data flow structure will invalidate the currently cached data structures, and our framework implementation detects this automatically.

2.4 Visualization Component
The Visualization Module is the component within the framework responsible for rendering a visual representation of the cached intermediate data produced by the processing module. The framework currently supports visualization modules implemented using any one of the available Java UI toolkits such as SWT or Swing, or advanced hardware accelerated native rendering using OpenGL. The visualization modules also monitor user interactions via mouse or keyboard which can be interpreted to perform view transformations.

In addition, every visualization module uses the concept of a ContentProvider to define an interface to the cached data structure. The visualization module makes modifications to a set of input parameters that are sent to the content provider. The content provider interprets these changes and retrieves the appropriate information required for rendering from the available intermediate data, notifying the rendering component of the visualization module when the new render data is ready. Every visualization component uses one ContentProvider which can employ multiple mappings into one or more intermediate data structures.

2.5 Framework Editor
We have also implemented an interactive editor, shown in Figure 4, that allows users to build and customize visualization methods using any valid combination of existing framework components. The editor operates much like a visual programming environment for data transformation, processing, and visualization. In the figure, datasources are placed on the left (green), and the directional connections depict the flow of data through an appendor and a look-up table operator (blue) to the processing module (yellow). Additionally, three visualization modules are mapped to the intermediate output of the processing module. When the customized application is executed, all three visualizations will be launched and linked to a set of common controls, in this case a timeline and a legend.
3 Network Visualization Methods

A variety of methods have been employed in the visualization of large datasets such as network data [3]. The purpose of these methods is to visually represent information containing a large number of entities, since human cognition supports a higher rate of comprehension of visual information when compared with tabular display [4], [5]. We will employ general network visualization methods which include various plotting and graphing methods of network performance metrics over time, as well as more complex visualization methods which account for the transactional nature of network data. Our visualization framework is designed to incorporate any of these methods, as well as methods developed within domains other than network analysis.

General visualization methods for network data include traditional graphs and plots depicting variation of network performance metrics over time. Metrics can vary from overall measures like bandwidth usage to more specific metrics like call completion rate (CCR) for transmissions over lossy networks. Simple graphs and plots are highly effective for most network metrics because they can be intuitively understood by a large audience with very little explanation.

More advanced techniques for visualizing network data have recently been proposed, most notably recent work by Daniel Keim [6]. The radial traffic analyzer, shown in Figure 5, is designed to display the relative bandwidth use (i.e. percentage of whole) per IP address and transmission protocol. In general, this visualization method is designed to support quantitative analysis of hierarchically structured datasets. Therefore while this visualization method is well-suited for visual analysis of network data, it can also be useful for any datasets that exhibit hierarchical relationships. In this example, the inner two rings of the radial graph depict the source and destination of transmissions over the network, respectively, designated by IP address. The outer two rings of the graph depict the source and destination ports of the transmissions. The ports used for each transmission identify a transmission protocol, such as HTTP or FTP.

3.1 Information Overview

To the data analyst, the quality and quantity of information conveyed by a visualization method is of greater importance than the implementations details of the method itself. Indeed the science of information visualization has been treated as an independent field of study since Edward Tufte began exploring these concepts without the use of digital processing or display [7]. We propose that the information value of a visualization method can be increased when correlated and co-located with complementary visualization methods, and that the framework we have developed enhances the value of the individual visualizations by promoting interactive visual correlation. We
will now discuss the information value of each visualization technique used in our example.

**Line Graph**
The line graph is designed to visually depict changes in value within a given dimension over time. Each identified group is assigned a unique color. We include a subtle fill effect beneath each line in the graph so that the viewer has awareness of larger values that may be off-screen. For our network analysis, this graph provides an intuitive display of bandwidth usage for each protocol on the network. Because of the nature of the line graph display, the user can easily conduct a comparative analysis of bandwidth usage on the network by individual protocol, assuming that the total number of protocols in the dataset is not large.

**Stacked Bar Graph**
The stacked bar graph complements the line graph visualization by displaying overall bandwidth usage on the network over time. Because the protocol groups are stacked upon one another in the graph, the user is primarily given information about the total bandwidth usage for each time period instead of bandwidth usage by protocol. While the protocols are still distinguished using unique colors, it is difficult to compare values between dimensions or to intuitively comprehend the quantitative value of each protocol group since the group is located on top of another group. We allow the user to rearrange the stacking order of the protocol groups in our implementation and visually query the quantitative values using mouse interactions. This graph is designed to support visual comprehension of overall bandwidth usage instead of comparative analysis among protocols, and therefore is well-suited to complement the line graph visualization.

**Radial Traffic Analyzer**
The radial traffic analyzer is a relatively new visualization method designed specifically to support network traffic analysis [6], and has been used to help monitor and interpret network intrusion attacks [8]. This method uses concentric rings of increasing diameter to visually depict a hierarchical relationship between multiple dimensions. For example, each unique segment within the inner ring corresponds with a top-level node of the data. Each node’s group of children will be arranged within the same arc of the next outer ring corresponding to the parent node’s location in the inner ring. For our network analysis, we have configured the radial traffic analyzer to display three rings, the first two representing the source and destination of each transmission (using an IP addresses) and the outer ring representing the transmission protocol used. This visualization provides quantitative information about the senders and receivers of data across the network, grouped by protocol. Our implementation of the radial traffic analyzer has been configured to show this information relative to a primary dimension of data common to the other two displays – protocol – and therefore this visualization is a suitable complement to the group.

4 Implementation
In this section we present our implementation of a custom multi-view network visualization application.
using the data visualization framework. We will also discuss how the user can interact with the customized application to perform knowledge discovery and engage in data discourse.

4.1 Interface Overview
The result of executing the customized visualization method from Figure 4 is shown in Figure 6. We can see the line graph, bar graph, and radial traffic analyzer views of bandwidth usage over time within a sample network dataset. The application provides two common controls, a timeline (bottom) and a legend (right). Each visualization individually monitors and responds to user interactions with these controls. The timeline control displays the current valid range of data in the time dimension and allows the user to indicate a range of data for all views to display as well as the current resolution to display, since our processing modules provide data at a variety of resolutions. The legend control is used to display unique entities with the data and the user can reorder, group, hide, or change the color associated with any of these entities. In this example, the legend displays the colors associated with each unique protocol and IP address in the dataset. Additionally, selection within the legend enables focus effects within each visualization. For example, when a protocol is selected in the legend, any data elements in either of the three displays that do not reference the selected protocol are faded out. All views support a query tool that allows the user to get more detailed quantitative information by hovering or clicking on appropriate areas of the figure.

5 Results
By presenting multiple linked views of data when performing visual analysis we can increase the quantity of information available to a data analyst. In our example we were able to provide the analyst with a much broader range of visual information about a network dataset captured during a simulation event not by developing a complex new visualization method but by complementing simple visualization methods with other techniques. Using our extensible framework, we have built a custom network visualization application which employs a variety of data visualization techniques to provide information on the total bandwidth usage and individual bandwidth usage by protocol over time, as well as a visual representation of which nodes on the network were contributing the most network activity during that time. All visualizations are linked to a common legend and to a timeline that provides the ability to view processed data at a variety of resolutions. This interactive information display is currently being prepared for use in within the Operational Test Command at Ft. Hood, Texas, and will improve the ability for analysts to provide decision-makers involved in the military acquisition process with more analytic information about the simulated performance of developmental systems.

6 Future Work
We are currently engaged in interaction analysis to improve the usability of the interactive editor, which will help us better organize the many functional components currently in development to support advance network analysis and other visual data analysis tasks. We will be implementing components to perform statistical analysis and interactive visualization of reliability and maintainability data. We would also like to continue to explore applying existing methods to datasets from new domains, as they become available. Additional future work will focus on performance improvements within the framework to support handling of very large datasets by integrating view-dependant data retrieval and distributed processing functionality.

7 Conclusion
By combining both general and domain-specific visualization methods whose information values complement each other we can enhance the ability of the analyst to engage in true discourse with data. The software framework we have developed allows users to interactively assemble high-level data processing and visualization capabilities to create custom data visualization applications. By combining complementary visualization techniques we can increase the amount of information available to a data analyst without increasing the complexity of individual visualization methods. The products of this research and development model will enable analysts from many domains, including military and homeland security, to detect the expected and discover the unexpected in diverse datasets [1].

References


