Proceedings of the 2014 Winter Simulation Conference
A. Tolk, S. D. Diallo, I. O. Ryzhov, L. Yilmaz, S. Buckley, and J. A. Miller, eds.

DATA VISUALIZATION FOR NETWORK SIMULATIONS

Christopher S. Main
L. Felipe Perrone
Greg L. Schrock

Department of Computer Science
Bucknell University
Lewisburg, PA 17837, USA

ABSTRACT

As many other kinds of simulation experiments, simulations of computer networks tend to generate high volumes of output data. While the collection and the statistical processing of these data are challenges in and of themselves, creating meaningful visualizations from them is as much an art as it is a science. A sophisticated body of knowledge in information design and data visualization has been developed and continues to evolve. However, many of the visualizations created by the network simulation community tend to be less than optimal at creating compelling, informative narratives from experimental output data. The primary contribution of this paper is to explore some of the design dimensions in visualization and some advances in the field that are applicable to network simulation. We also discuss developments in the creation of the visualization subsystem in the Simulation Automation Framework for Experiments (SAFE) in the context of best practices for data visualization.

1 INTRODUCTION

As scientists, we have long acknowledged that data are the primary enabler of discovery and knowledge. Empirical data, however, don't speak for themselves even though they may carry a wealth of information. The rigorous observer must expend time and energy sifting through collected data in order to be able to identify the "golden nuggets" that lead to entirely new realizations or to the (in)validation of hypotheses.

Advances in computing technology for the generation, the collection, and the storage of data have enabled us to amass unprecedented volumes of experimental samples. This is particularly true of computational sciences, such as simulation, which can now generate data faster and in larger amounts than it has ever been possible. This blessing, however, turns into curse when one considers the added difficulties in finding meaning in data. The stories that data may be able to tell can be either clearly and unambiguously visible or they may be hidden in proverbial haystacks.

The skillful filtering, transformation, and visual encoding of the data can make a world of a difference in creating graphics that carry messages which are compelling and easy to understand and internalize. Data visualization encompasses these competences, but this area is vast and comprises various interdisciplinary specialties such as cognitive psychology, visual perception, aesthetics, design, and algorithms. It is not our goal in this paper to investigate algorithms for data transformation, to discuss aesthetics extensively, or to go deep into studies of visual perception. Our intent is to bring to light to some of the important considerations in data visualization, to explore some of the options for telling the stories hidden in the data of network simulations, and to present our approach to build visualization functionality into a framework to support network simulation.

2 BACKGROUND

Research is the field of data visualization spans a variety of disciplines including, but not limited to, graphic design, statistics, visual perception, cognitive psychology, and computer science. When building visualizations to communicate our experiment results, we are most often oblivious to the body of work in this field and sometimes fail to observe best practices. It is unreasonable to expect that each of us would spend the substantial time to develop this expertise just so that we would build a few graphics for our publications. It is quite reasonable to expect, however, that the construction of the software tools that support our endeavor would incorporate this body of knowledge so as to shield us from the details and the complexities in visualization. In this section, we call attention to some of the fundamentals in the field with the aim of informing fledgeling designers of visualizations for simulation results.

Semiology of Graphics (Bertin 2010) is one of the most important and influential works ever published in the field of information graphics. Originally published in French in 1967, Bertin's work paved the way for the fields of exploratory data analysis and information visualization. A cartographer by trade, Bertin developed an insightful theory of graphics almost entirely through his own perception. Bertin's early work has inspired much research in the field of information graphics and his work has held up in modern studies of cognition and perception (Garlandini and Fabrikant 2009). According to Bertin, the purposes of graphic representations are to record, communicate and to process information. He remarks that to enable the best cognitive processing of scientific data presented visually, one should strive for *"reducing the comprehensive, nonmemorizeable inventory to a simplified, memorizable message."* Effective information graphics are among the best methods to disseminate meaningful messages that emerge from experimental data. The human faculty of vision, which is the first stage in processing the message contained in the information graphic, is subject to six retinal variables (Bertin 2010):

- **Size** describes a variation in the dimensions of a mark on a plot which causes perceptual stimulus. This variable applies to points, lines, and areas. An important consideration about size variation is that it is *dissociative*, meaning that it will dominate any other retinal variable with which it is combined.
- **Value** variation describes the ratio between black and white objects on a given surface. An order exists within value variation, from light to dark, and not following this ordering will produce a graphic that is not suitable for visual interpretation. Interestingly, value is also dissociative and not quantitative by itself. When combined with size variation though, value variation can be used to represent quantitative data.
- **Texture** describes the number of distinct marks or symbols within a certain area. Examples of common texture elements include arrays of circles, squares, or lines. The creator of a graphic needs to take care when using textures as they can cause uncomfortable visual sensations. This effect is commonly referred to as *moiré vibration* (Tufte 2001).
- **Color** variation refers to changes that can be perceived between identical areas which have the same value. There are many aspects of utilizing color variation which can trip up a designer, such as accidentally creating an association between color variation and order where one does not exist in the data.
- **Orientation** is "the difference in angle between fields created by several parallel signs". For orientation variation to create visual stimulus, the shapes must be linear. Furthermore, the designer of the graphic cannot exploit too many variations on orientation. There must be easily recognizable categories of variation for them to have visual meaning.
- **Shape** describes the variation in appearance of objects of equal sized area. Obvious examples of shapes are circles, squares, and triangles, but countless others exist. In certain types of graphics, such as maps, the use of *mimetic shapes*, or shapes that imitate what they represent, can make a graphic much easier to comprehend. It is important, though, not to use mimetic shapes where the mark is not intended to have any particular meaning.

These variables represent variations in perception that occur above the plane of the graphic and are often discussed in the context of experimental psychology as defining how humans perceive depth. These variables are all the tools that a designer can use in the creation of two-dimensional graphics and therefore importance of understanding how to manipulate them is paramount. The effectiveness of the graphic as a communication tool depends on the designer's adroitness at applying the rules of syntax and semantics of visual language. Therein lies a problem, as most of us haven't received formal training in this specialized language and, yet, we use a variety of powerful tools to build graphics that mean to tell stories about our experimental data. The tools allow us to manipulate the retinal variables to our hearts' content, but rarely (if ever) make efforts to protect us from our mistakes.

Although it is beyond the scope of this paper to discuss all the common mistakes and best practices associated with the manipulation of retinal variables, for the sake of illustration, we discuss the cases of *texture* (briefly) and *color* (in more detail). When color is not available for the creation of a graphic, designers may inadvertently resort to *fill patterns* so as to use texture in distinguishing different elements of the graphic. This practice is intuitive, but not recommended, as it can cause dizziness to the viewer, particularly when the "patterns consist of vivid lines running in various directions" (Few 2012). Figure 1 exemplifies the effect of moiré vibration in a bar chart.

Arguably, the most familiar misuses of a retinal variable in common experience are associated with color. Most of us can relate experiences in which elements of a graphic are either hard, if not impossible, to view due to the poor choice of color. A typical example is the use of neon green over a white background, a default used in plotting with the popular Unix tool gnuplot, illustrated also in Figure 1. Although the combination leads to somewhat acceptable results on a modern display screen, it is very hard to see when projected, a fact which many have discovered only in the middle of a presentation. Using color to create distinctions between elements of the graphic, however, is only one application of this retinal variable.

To understand well how color variation can be used in the construction of graphics, one must first understand its three fundamental attributes. *Hue* corresponds to the wavelength of the color or to our common understanding of color; for instance: violet, blue, green, yellow, orange, red, purple, and gray. *Value* (or lightness) corresponds to the percentage of black in the corresponding shade of gray. *Saturation* corresponds to the degree of "purity" in a color, that is, its distribution across the spectrum of wavelengths. A pure or saturated tone is one that involves no mixture with other colors. Such a tone is neither darkened by the addition of black, or lightened by the addition of white. Constant value tones, on the other hand, are produced by the addition of some black or some white. Saturated tones are not of constant value; rather, they vary in value according to the color. Figure 2 illustrate the difference between hue and value.

The difference between saturated tones and constant value tones leads to two distinct uses of color in information graphics. Saturated tones can convey the representation of *order* among numerical values represented in a graphic. Yellow denotes the smallest value and either purple or violet represents the largest value. The values in between yellow and purple should be orange and red, in that order. This ordering is referred to as the "warm" tones. The colors between yellow and violet should be green and blue, in that order. This ordering is referred to as the "cool" tones. One should not mix warm and cool tones when denoting order, as it creates visual confusion. If one does not want color to be associated with ordered values, then constant value tones should be used.

When using constant value tones it is important to maintain a level of selectivity, or "distinguishability," between colors. Selectivity is maximal near saturated tones and diminishes as the tone moves towards either white or black. However, there are tones that are more selective than others depending on the tone's value. For example, with light value tones, blue, purple, violet, and red tend to look grayish and should not be used. It is advisable to choose steps around yellow, from green to orange.

While color variation is an effective technique for differentiation, it is not without problems. Individuals with disabilities in chromatic perception, such as color blindness, may perceive little to no information in a graphic constructed with color. Reproducing color on paper can also be difficult, and color graphics when reproduced monochromatically may have little value. One possible way to mitigate these disadvantages is

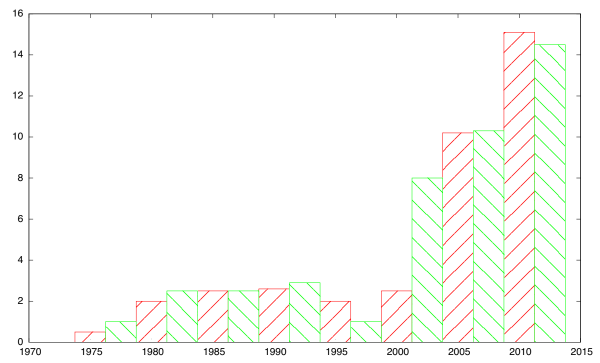
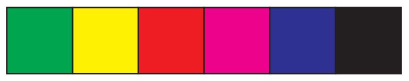
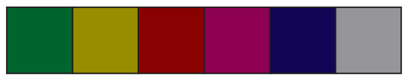


Figure 1: Moiré vibration in a bar chart.



(a) Saturated tones to illustrate *color hue*.



(b) "Even colors" to illustrate *color value*.

Figure 2: Difference between hue and value.

to combine color with another variable, such as size or shape. Most importantly, Zelazny (2001) reminds designers that the use of color should not be driven by its "decorative" aspect. Instead, he argues, color should be used purposefully as a means to help one tell the stories that lie behind data.

It should be noted that the process of designing information graphics doesn't start "from the bottom," that is, from considerations of the perceptual impact of retinal variables. The design should start "from the top" by identifying which relationships within the data are important to expose. The nature of the data and message it carries are important dimensions in this design process and have given rise to various related though different categorization schemes. Zelazny (2001) proposes a taxonomy based on the overarching purpose of the graphic: component comparison, item comparison, time-series comparison, frequency distribution comparison, and correlation comparison. Few (2012) renames some of these these and incorporates them into an augmented taxonomy of relationships: part-to-whole, ranking, time series, correlation, deviation, geospatial, nominal comparison. Börner and Polley (2014) propose categories derived from the answers that graphics can provide to the questions one might want to ask of data: "WHEN" (temporal data), "WHERE" (geospatial data), "WHAT" (topical data), and "WITH WHOM" (relational structure data). These categorization schemes are tools for reasoning through the initial stages of the design process and it can be argued that there isn't one that is necessarily better than the others. What is important is that the selected visualization graphic agrees with the nature of the data and is effective in support of recording information, conveying meaning, facilitating discovery, and supporting perceptual inference, among other goals (Meirelles 2013).

Lastly, we ought to keep in mind that although each scientific community might have a historic predilection for specific types of graphics, there is ample room for creativity and experimentation with different alternatives. Although there is value in the use of the common visual language on one's métier, data visualization is as much art as it is science and it is subject to continuous evolution. In the next section, we discuss some of the common "visual idioms" in the field of computer network simulation and suggest directions for experimentation.

3 USES OF VISUALIZATION IN NETWORK SIMULATION

3.1 Time-Oriented Data

In the analysis of many types of processes, we are often interested in tracking how a given metric evolves over time. To that end, we collect a sequence of samples of the metric at successive times, either regularly or irregularly spaced. This data may be graphed as a *scatter plots* or *line graphs*, where the value of the metric is mapped to the y-axis and time (the independent variable) to the x-axis. The representation shows values in the order in which they arose and allows one to observe the dependency between successive values. According to Tufte (2001), *time series* graphics are most appropriate for "big data sets with real variability," but may be constrained to serve as descriptive chronology without causal explanation. Since time series graphics are extensively used in the analysis of real networks, it is not surprising that their use is also prevalent in network simulation. Because plotting time series requires basic functionality of graphing software, general purpose tools such as spreadsheets (MS Excel, Apple Numbers, gnumeric), gnuplot, and *The R Project for Statistical Computing* are equipped for them. In addition to these, powerful tools for network analysis such as RRDtool (Oetiker 2013) can also be leveraged.

As Aigner et al. (2008) observe, time-oriented data sets tend to be large both in terms of the number of data items and also in terms of the number of observed attributes. Regardless of the data having come from physical measurements or from simulation, there exists considerable likelihood that visualizations time-oriented data may end up in overcrowded displays that hinder the comprehension of the data. For this reason, it is important that analytical methods for data abstraction are used to pre-process the data and counter the overcrowding effect. The authors remark that "visualization frameworks have not yet considered time as a special dimension but rather as a common quantitative parameter" and go on to point out three different criteria for considering the relationship between data and time in the context of visualization, discussed as follows.

First, one ought to consider whether time is used simply to define an ordering for data points (linear time) or to represent the epochs that repeat themselves in cyclic successions (cyclic time). Although the standard time series graphics are adequate for dealing with visualizations of linear time, they are not effective for cyclic time because they make repetitive quantitative patterns difficulty to identify. Aigner et al. (2008) call our attention to the possibilities opened by plotting data series in cyclic time on graphics where the time axis is shaped as a spiral. This visual design known as *Spiral Graph* aims to enable the detection of previously unknown periodic behavior of that data and requires the definition of a cycle length parameter. The authors recommend the application of animation or interaction with the user to facilitate finding the appropriate setting of the cycle length. In the analysis of data produced by running computer networks and by their simulation, it is quite common for one to be interested in identifying trends that emerge in cyclic time. For instance, when observing traffic through a network node, we might want to see what happens on the tenth hour of each day over a succession of various days. Although not commonly used in reports of network performance analysis using data from physical systems or from simulation, Spiral Graphs are clearly applicable in this context. (Time series data from networks go through cycles determined by the succession of hours within a day, of days within a month, etc.) See this for an interesting example of interactive Spiral Graph.

A second criterion for viewing the relationship between time and data regards the tension between dealing with time as either *points* or *intervals*. Data collected from the simulation at specific points in time represent values sampled particular settings of the simulation clock. As such, these data can only speak of particulars states in the trajectory of a metric. Time interval data, on the other hand, is represented by some kind of mathematical function applied to two or more values in the trajectory. As such, time interval data is represented by summary statistics, or *data summaries*, such as the mean or the standard deviation of the collection of observations within a well defined time period. Traditionally, graphics of both time point and time interval data in network simulation have appeared as either scatter plots or line plots. These two types of plots may be considered adequate for time point data, but rigorous reporting in the case of time interval data requires the display of more than just point estimates for the averages in each interval. In general, there exist several mechanisms for visualizing summary statistics when plotting *any* two variables (not just time intervals on the x-axis). One of the simplest options is the use of *confidence intervals* around point estimates. Even though the functionality to that end is offered by most graphing software, in network simulation, quite often authors don't use confidence intervals, as reported by Kurkowski et al. (2005). Other visualization techniques for data summaries include *box-and-whiskers* plots (Tukey 1977), and *quartile* and *parallel schematic* plots (Tufte 2001). These options provide the viewer with more information on the distribution of data without excessive clutter. Regardless of the graphing technique, the designer should keep in mind that there is an inherent limitation in the use of data summaries: they tend to obscure unusual characteristics that may be present in the data. For that reason, the visual exploration of data shouldn't be limited to the use of statistical summaries.

Another technique that can be useful for visualizing time interval data is the *stacked graph*, which creates different layers for the multiple time series to plot and places them on top of each other. Byron and Wattenberg (2008) discusses the evolution of stacked graphs. The authors remark that the popular use of this category of graphs indicates the hypothesis that "stacked graphs have an ability to communicate large amounts of data to the general public in an intriguing and satifactory way" and argues that their value extends also to expert analysts. Given the nature and the volume of data produced by network simulations, it is likely that stacked graphs might be a good alternative to traditional graphs in creating compelling and easy to understand visual narratives. There are challenges in the construction of stacked graphs, however, related to perceptual variables such as geometry, color, layer ordering and labeling, which have been addressed in the *Streamgraphs* design introduced by Byron and Wattenberg (2008). Tools for plotting Streamgraphs are evolving and existing options support the development of programs in R, Python, Java, and Javascript (with the D3 library). See the following examples for illustrations of stacked graph and Streamgraph.

The third and last characterization of types of relationships between data and time defines two dimensions of interest to simulation. In the *ordered time* domain, the visualization works with sequences in which each element relates to others by a "happens before" relation. This domain is applicable to simulation, in general, for expressing the ordering of events and could be particularly helpful in understanding causality chains. The ability to represent the events of a network simulation on a timeline would enable the validation of protocol models by visual inspection. Nevertheless, it is particularly difficult to represent long sequences of events with multiple, concurrent causality chains without overwhelming the viewer. For this reason, this type of visualization is more suitable for interactive explorations of the data. Interactivity would also be helpful for visualizing the *multiple perspectives* domain, in which a user would be able to compare multiple views portraying time-dependent variables and their relation to a single time axis.

In this section, we illustrated a small fraction of the wealth of different options that exist for visualizing *time-oriented data*. Future work in the visualization of network simulation data should attempt to investigate how effective these options are in this application domain.

3.2 Geospatial Data

The uses of geospatial data in network simulation can be multiple and relate to the necessity of identifying the location of a piece of information within a given landscape. One obvious use of this type of data includes the visualization of *topographic maps* for the modeling of space in wireless network simulation. These are not visualizations of the output produced by simulations, but rather of model inputs. There exist multiple methods for visualizing the various formats of digital elevation maps used for modeling space and discussing them goes beyond the scope of this paper.

A form of geospatial data that is generated dynamically in the simulation of wireless networks are *heath maps* or *isopleth maps* to represent the incident signal strength or signal-to-interference ratio on receivers positioned on a model of two-dimensional space. Heat maps for this type of data may be animated to show dynamically how received signal strength evolves as network nodes adjust their transmission power and/or move about in space. The use of heat maps, however, is not constrained to expressing the relationship between the value of a metric and a position in space. An example of this is provided by Bosaw (2010), who uses a heat map to express the difference between packet error rates computed for the same wireless network scenario by two different simulators. When implemented correctly, this type of visualization can become a valuable tool to promote the understanding of simulation results. Although heat maps do not match the precision of other representations, they are effective in packing a substantial amount of information into a limited area (Few 2013). Static heat maps are easily built with a variety of programs such as Microsoft Excel, gnuplot, and R Statistics. In spite of these benefits, the use of heat maps is fraught with peril. Since they tend to place a multiplicity of different colors on the same graphic, heat maps can lead to misinterpretations due to the limitations of human color perception. This can be a hindrance for individuals with varying degrees of color blindness, as a substantial part of the information is encoded as color. From a different disciplinary domain, Bojko (2009) proposes transferrable guidelines for the use heat maps.

3.3 Relational Structure Data

Data on relational structure speaks of arrangements of the vertices (nodes) and the edges (links) that constitute a graph, which is the most fitting representation of a network. Nodes in a graph can be all of the same type (as in a *unipartite graph*) or of different types (as in a *multipartite graph*). Links represent interactions or relationships between nodes and may also have varying types. Undirected links model mutual connections, where as directed links model unidirectional connections. Additionally, links may be unweighted (indicating only the connection between two vertices) or weighted (indicating an additional attribute that further qualifies the connection).

The conceptual simplicity of the graph as a construct, however, does not imply that it is always easy to represent graphically. The *node–link* representation uses symbols to represent nodes and lines to represent links. There are two main challenges in creating readable visualizations of this representation as the number of nodes and links grows: it becomes increasingly more difficult to avoid the occlusion of nodes and the crossing of links. When visualizations include additional graph theoretical attributes such as link directionality and weight, the problem gets even harder. Drawing the graph for the sake of providing instantaneous visual cues about metrics such as connectedness and node clustering, complicates the visualization even further, but it gets worse. Quite often, additional graph theoretic metrics need to be included in the graphic, as well as identifications for specific paths, links, and nodes, distinct colors for graph components, etc. This augmented scope for the visual representation of relational structure data defines the challenges encountered in creating visualizations of communication and computer networks.

Various tools currently exist for the visualization of network simulations; examples include iNSpect (Kurkowski et al. 2005), (Abraham and Riley 2014), and various others described by Musznicki and Zwierzykowski (2012). The state of the art in these visualizations for network simulation, however, has yet to evolve toward the expressiveness of modern representations described by Meirelles (2013).

4 VISUALIZATION IN SAFE

Having presented a formal background for the construction of effective visualizations in the previous sections, we now focus on a development of our own. The *Simulation Automation Framework for Experiments* (SAFE) (Perrone, Main, and Ward 2012) is being developed to guide novice and experienced users of the *ns-3* network simulator along a rigorous empirical methodology. SAFE's overarching goal is to provide a comprehensive environment that makes it easier to stage simulation experiments that produce more credible results. This framework provides users with a mechanism to create both interactive and static visualizations of simulation results that is based on the subsystem with architecture depicted in Figure 3. SAFE is based on an application model in which the experimenter works on a client computer and interacts with a remote server either via a command-line interface or via a web-based interface accessible with a standard browser. After an experiment is configured and deployed on simulation *worker* machines, simulation output data is sent back to the server, which stores them in a database for persistence.

SAFE's visualization subsystem comprises cooperating components that execute in the server and in client computers; it is described in detail by Main (2013). The server offers an API to a visualization backbone that serves as the gateway for remote access to two components: a database that stores the complete experimental setup and simulation results, and an analysis module. The latter implements functionalities such as performing transformations on the experiments' results to support the needs of visualization components that execute remotely.

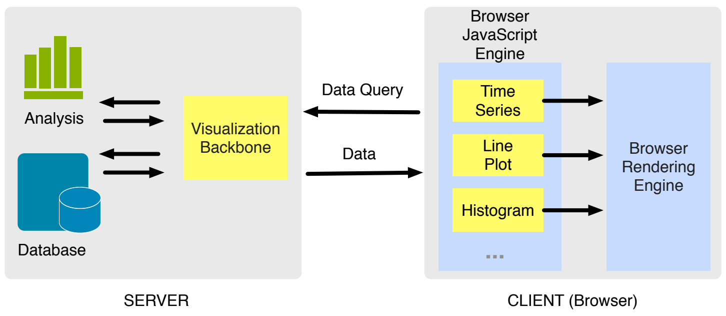


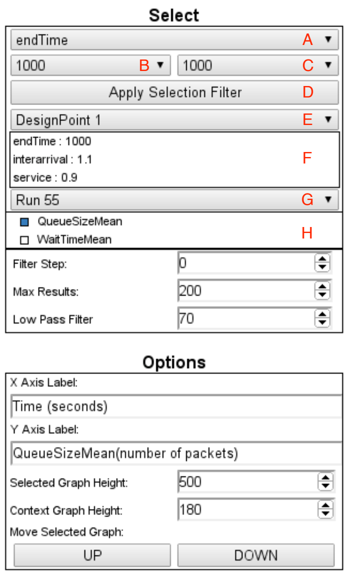
Figure 3: Support for data visualization in the architecture of SAFE.

Using a web browser on a remote client, the user accesses the visualization system's URL in the server, which returns a dialog for the selection of the type of visualization. Once a specific type of visualization is chosen, the browser is served a minimal base of HTML code, which consists of one or two tags that indicate where JavaScript code will add data visualization and control elements of the interface. Within the Javascript code, we invoke functions in the *D3.js* library to select and to insert additional components into the the web page. CSS files define the formatting of most of the components in the visualization modules, while the attributes of the remaining components are defined in JavaScript.

Each visualization is presented with a "control panel" similar to the one shown in Figure 4, which allows the user to specify what will be displayed and to customize the appearance of a plot. The control panel is implemented with *Asynchronous JavaScript and XML* (AJAX) so that different elements (buttons, pull down lists, text entry boxes, etc.) can react dynamically to user input. The example shown in this figure includes several elements worthy of note. **A** is a pull down list that contains all the factors of the selected experiment so that the user can select one to explore visually. **B** and **C** are pull down lists to allow the user to select the respective lower and upper limits of a range of levels assigned to the chosen factor in this experiment. The items in these three pull down lists are retrieved from the experiment's configuration recorded in the database. Once selections are made in these three lists, the user presses the *Apply Selection Filter* button (**D**), which creates a database query to retrieve all the experimental design points in which the chosen factor is assigned levels in range [**B**, **C**]. The result of this query defines the contents of the pull down list of design points in **E**, from which the user selects one specific design point. This selection issues another database query which returns the complete description of the design point displayed in **F**. Another query is used to populate the list in **G**, which shows the identifications of all independent simulation runs made for this same design point with different random number generator seeds. **H** displays a check box for each metric recorded in the simulation run; once the user selects a check box, the correspondent data series is plotted in the visualization window.

Other elements that may appear in the "Select" portion of the control panel allow the user fine grained control over the data passed to the plotting code. In this example: "Filter Step" defines whether all points in the data series are displayed, or every other point, every third point, etc.; "Max Results" defines the total number of points that will be plotted; "Low Pass Filter" defines the parameter for the application of a sliding window average applied across the data series. The manipulation of result data is performed not in the JavaScript code in the client's browser, but instead on the server's analysis module. The elements in the client's control panel invoke these functionalities via the API offered by the server's visualization backbone. The "Options" portion of the control panel serves to customize the dimensions of the plot and also the labels on the axes. The labels are pre-populated with the results of a database query to retrieve from the experimental set up the descriptive text for the metric on each axis, as well as the associated unit of measure.

Adjacent to the control panel, the user's browser displays the chosen visualization component. The example in Figure 5 shows a *time series* visualization that embodies the Visual Information Seeking Mantra due to Shneiderman (1996): *"Overview first, zoom and filter, then details-on-demand,"* which we dissect



Select

endTime ▾	A
1000 ▾ (B)	1000 ▾ (C)
Apply Selection Filter	D
DesignPoint 1 ▾	E
endTime : 1000 interarrival : 1.1 service : 0.9	F
Run 55 ▾	G
■ QueueSizeMean ☐ WaitTimeMean	H
Filter Step:	0
Max Results:	200
Low Pass Filter	70

Options

X Axis Label:	Time (seconds)
Y Axis Label:	QueueSizeMean(number of packets)
Selected Graph Height:	500
Context Graph Height:	180
Move Selected Graph:	UP / DOWN

Figure 4: Control panel.

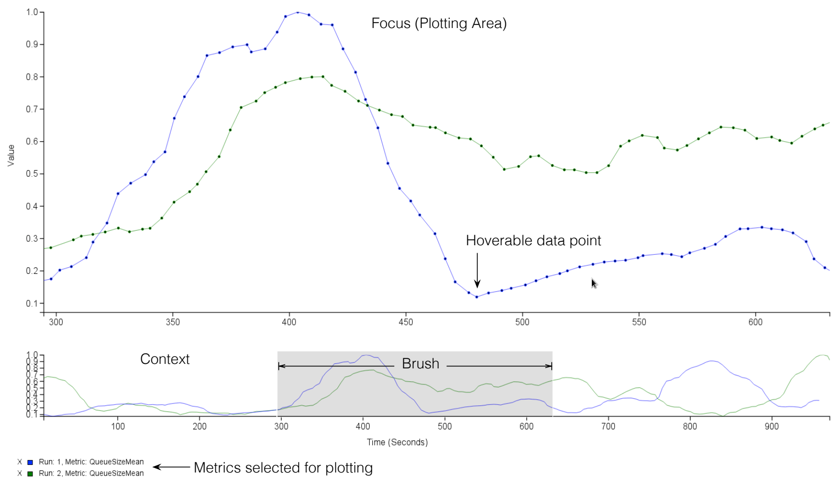


Figure 5: Micro/Macro visualization in SAFE.

as follows. The main goal of the *overview* is to give the viewer the ability to make quick sense of the entire data series. The overview generally contains an adjustable area that can be used to select an portion to expand in detail by *zooming*, which focuses in on selected items of interest within a collection. Making this action happen gradually allows the user to retain a sense of position within the data, which is important in facilitating interpretation. To *filter* is to remove items that are not of interest. The concept of *details-on-demand* corresponds to having a user request more details about an item or group of items, but only if and when they are needed. For this to happen, the collection of items needs to be trimmed down to a reasonable size so that individual items can be selected, which can be accomplished by clicking or hovering on the item. Selection results in a popup element that provides the value of the metric associated with the item. The *Focus* or *Plotting Area* shown in Figure 5 displays a curve on which dots indicate a selection of hoverable data points. Since the volume of available data could easily overwhelm and clutter the focus, we apply a heuristic for adjusting dynamically the density of hoverable points – see Main (2013) for details.

Schneiderman's principle gave rise to the notion of *micro/macro visualization*, which provides an overall picture of the data while also providing a deeper level of detail upon closer inspection. The general idea behind this design is to expose larger trends and let the viewer determine where to focus on details. Micro/macro visualizations do not make viewers rely on visual memory to make comparisons or choices, as a graph of the complete data series is always displayed. This gives the viewer a better chance of comprehending the data and discovering interesting relationships within them (Tufte 1990). Figure 5 illustrates how SAFE's visualization for time series uses this design: the bottom part of the graphic (labeled *Context*) shows the entire data series and corresponds to the macro view. The upper part of the graphic, enclosed within the x- and y-axes (labeled *Focus*), shows the expanded view of a segment of the whole data series.

An important aspect of the visualization modules in SAFE is *interactivity*. Wilkinson and Wills (2005) define two categories of exploratory controllers for interactivity: indirect manipulation tools and direct manipulation tools. Indirect manipulation tools operate on an alternative representation of the graphic and not the graphic itself. Examples of indirect manipulation tools include sliders, buttons, and joysticks. Direct manipulation tools, which are generally preferred, operate on the graphic itself. They provide the benefit of the user not having to look away from the graphic when it is being manipulated. A specific method for direct manipulation that has earned significant attention is *brushing*. The action corresponds to sliding a "window," that is, a graphical device called *brush*, over a certain region of the graphic to highlight a collection of objects. As the width of the brush is adjusted and as the brush is moved over the context, the portion of the graphic detailed on the focus is dynamically adjusted. Other visualization modules in SAFE, such as those for *line plots* and *scatter plots*, are designed around the same principles of micro/macro visualization and interactive controls.

Another important aspect in the development of SAFE's interactive visualizations regards the combination, in the same plot, of two data series where each corresponds to a different metric generated by simulation. This type of scenario is sometimes addressed by the *double axes* plot design, in which left and right vertical axes exist, each with their own scale and possibly unit of measure. (In network simulation, users often express the desire to graph two very different metrics on two vertical axes which share one same horizontal axis denoting time.) Although intuition may indicate that the double axes design is helpful, the experts recommend against it, as the relative differences of the two scales can create a deceptive graphic (Wilkinson and Wills 2005, Wainer 1997, Wickham 2009). There do exist alternative designs that are better for comparing two different series which have different units or which have the same units but significantly different ranges. Wickham (2009) identifies two techniques for dealing with this problem. We discuss each one in turn as follows.

The first technique, called *normalization*, rescales independently each of the variables that would have been on different y-axes to a common range, such as $[0,1]$. Figure 6 shows a graphic where two variables, that have the same unit and possibly different scales, are plotted on a normalized y-axis. This type of design facilitates comparison by plotting the data series of two variables on the same graph and, at the same time, avoids having one of the curves be overwhelmed by the other when the range a data series is significantly larger. The second technique is to use *faceted graphics*, which place, on top of one another, multiple individual plots that share the same x-scale, but not necessarily the same y-scale. Figure 7 shows an example of how faceted graphics can be used to represent data series of different natures sharing a common base on the x-axis. Both techniques are available in SAFE, in combination with micro/macro visualization. When faceting is used, we impose no limit on the number of facets that can be created and provide *UP* and *DOWN* buttons in the control panel (see Figure 4) to allow an individual facet to be moved closer to another one for comparison.

5 FUTURE DIRECTIONS FOR VISUALIZATION IN SAFE

Work on SAFE's visualization subsystem is ongoing and its functionalities are being refined and expanded. Currently, the components that have been finalized support time series, line plots, and scatter plots. The development of these has served to validate and to refine the architecture of the visualization subsystem and of user interfaces. Additional developments planned for the near future include support for other common use cases in network simulation, discussed in the following.

The typical goals of network simulations commonly include the presentation of empirical probability distributions of data and the comparison between two or more distributions. There are numerous graphical designs for presenting distributions, some of the simplest being *discrete data histograms*, *continuous data histograms*, and the *box-and-whisker* plot, discussed by Tukey (1977) and Tufte (2001). A common method for comparing data distributions is the *quantile-quantile plot* (Q-Q plot). This type of graphic shows one distribution along the x-axis and the other along the y-axis. When the two distributions are similar, the points of the Q-Q plot fall along the $y = x$ line. Comparing data to theoretical distributions in this manner helps to determine if data fits a certain theoretical model (Tukey 1977, Cleveland 1993).

In addition to implementing visualization modules to support these use cases, we intend to explore the application of other designs such as stacked graphs, Spiral Graph, Streamgraph, in the context of network simulation data. Finally, we plan to investigate the applicability of the *small multiples* concept (Tufte 2001), which can illustrate how something evolves over time by showing small changes over various shrunken, high-density graphics arranged as a matrix. Small multiples facilitate comparison and is very effective at breaking down dense data into small, comprehensible units. Using small multiples visualization may

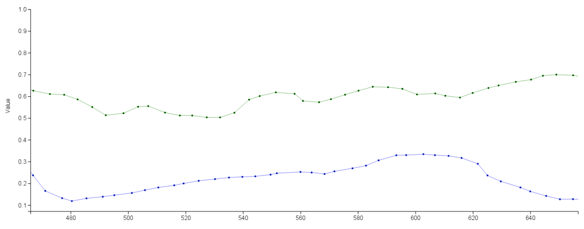


Figure 6: Normalization.

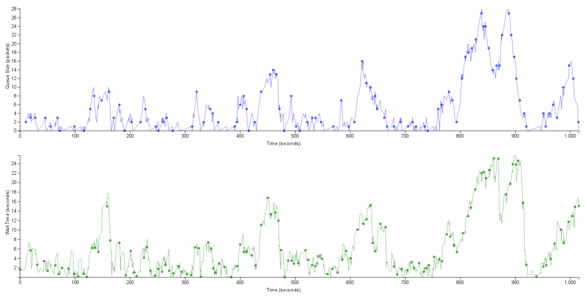


Figure 7: Faceting.

turn out to be helpful in different aspects of network simulation, such as in showing the evolution of a connectivity graph as network nodes roam the simulated space.

6 CONCLUSION

As simulationists, we frequently create graphics from the data our experiments generate. However, more often than not, our training does not include much knowledge from the field of visualization and our skills are developed through practice and experience, trial and sometimes much error. We hope that this paper provides the reader with the appreciation for the complexities of visualization and with incentives to learn more about a fascinating field. We hope, also, that the material in this paper stimulates the developers of visualization software for network simulations to consider incorporating into their work some of the best practices in information design and to explore modern visualization strategies. We do not advocate that we should all try to develop deep expertise in visualization. Instead, we expect that it would be highly beneficial for the field of simulation to meet with the minds of those who have the specialty we lack. Potentially, this could enhance significantly the impact of the lessons that can be learned from network simulation data.

ACKNOWLEDGMENTS

This development of SAFE is supported by the National Science Foundation (NSF) under grant numbers CNS-0958139, CNS-0958142, and CNS-0958015. Any opinions, findings, and conclusions or recommendations expressed in this material are the authors' alone and do not necessarily reflect the views of the NSF.

AUTHOR BIOGRAPHIES

CHRISTOPHER S. MAIN earned a B.S. in Computer Science from Bucknell University, in 2013. For one and a half years, he worked on the development of SAFE and made contributions to its experiment control and visualization components. He currently works for a Fortune 100 insurer. His email address is csm024@bucknell.edu.

L. FELIPE PERRONE is Associate Professor of Computer Science at Bucknell University. His interests include modeling and simulation, computer networks, and high performance computing. His email address is perrone@bucknell.edu.

GREG L. SCHROCK is a B.S. in Computer Science and Engineering student at Bucknell University (class of 2016). His work on SAFE focuses on the design and implementation of a web-based interface for experiment management and data visualization. His email address is gls022@bucknell.edu.

REFERENCES

Abraham, J., and G. F. Riley. 2014, May. *NetAnim*.

Aigner, W., S. Miksch, W. Muller, H. Schumann, and C. Tominski. 2008, Jan. "Visual Methods for Analyzing Time-Oriented Data". *IEEE Transactions on Visualization and Computer Graphics* 14 (1): 47–60.

Bertin, J. 2010. *Semiology of Graphics*. Esri Press.

Bojko, A. 2009. "Informative or Misleading? Heatmaps Deconstructed". In *Human-Computer Interation – New Trends*, edited by J. Jacko, Volume 5610 of *Lecture Notes in Computer Science*, 30–39: Springer-Verlag.

Börner, K., and D. E. Polley. 2014. *Visual Insights – A Practical Guide to Making Sense of Data*. MIT Press.

Bosaw, T. 2010. "An Improved 802.11b Interference Model for Network Simulation". Master's thesis, Dept. of Electrical Engineering, University of Washington.

Byron, L., and M. Wattenberg. 2008, November. "Stacked Graphs – Geometry & Aesthetics". *IEEE Transactions on Visualization and Computer Graphics* 14 (6): 1245–1252.

Cleveland, W. S. 1993. *Visualizing Data*. 1st ed. Murray Hill, N.J.: AT&T Bell Laboratories.

Few, S. 2012. *Show Me the Numbers – Designing Tables and Graphs to Enlighten*. 2nd ed. ed. Analytics Press.

Few, S. 2013. *Information Dashboard Design*. Analytics Press.

Garlandini, S., and S. I. Fabrikant. 2009. "Evaluating the effectiveness and efficiency of visual variables for geographic information visualization". In *Proceedings of the 9th international conference on Spatial information theory*, COSIT'09, 195–211. Berlin, Heidelberg: Springer-Verlag.

Kurkowski, S., T. Camp, and M. Colagrosso. 2005. "Manet Simulation Studies: The Incredibles". *ACM SIGMOBILE Mobile Computing and Communications Review* 9:50–61.

Kurkowski, S., T. Camp, N. Mushell, and M. Colagrosso. 2005, Sept. "A visualization and analysis tool for *ns-2* wireless simulations: iNSpect". In *Proceedings of the 13th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems*, 503–506.

Christopher S. Main 2013, May. Honors Thesis, Department of Computer Science, Bucknell University.

Meirelles, I. 2013. *Design for Information*. Rockport Publishers.

Musznicki, B., and P. Zwierzykowski. 2012, September. "Survey of Simulators for Wireless Sensor Networks". *International Journal of Grid and Distributed Computing* 5 (3).

Oetiker, T. 2013, June. *RRDtool Logging & Graphing*.

Perrone, L. F., C. S. Main, and B. C. Ward. 2012, December. "SAFE: Simulation Automation Framework for Experiments". In *Proceedings of the 2012 Winter Simulation Conference*, edited by C. Laroque, J. Himmelspach, R. Pasupathy, O. Rose, and A. M. Uhrmacher.

Shneiderman, B. 1996, September. "The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations". In *Proceedings of the 1996 IEEE Symposium on Visual Languages.*, 336–343.

Tufte, E. R. 1990. *Envisioning Information*. 1st ed. Cheshire, Conn.: Graphics Press.

Tufte, E. R. 2001. *The Visual Display of Quantitative Information*. 2nd ed. Cheshire, Conn.: Graphics Press.

Tukey, J. W. 1977. *Exploratory Data Analysis*. 1st ed. Addison-Wesley Series in Behavioral Science. Reading, Mass.: Addison-Wesley Pub. Co.

Wainer, H. 1997. *Visual Revelations: Graphical Tales of Fate and Deception from Napoleon Bonaparte to Ross Perot*. Copernicus.

Wickham, H. 2009. *ggplot2: Elegant Graphics for Data Analysis*. 1st ed. Use R! New York: Springer.

Wilkinson, L., and G. Wills. 2005. *The Grammar of Graphics*. 2nd ed. New York: Springer.

Zelazny, G. 2001. *Say It With Charts: The Executive's Guide to Visual Communication*. 4th ed. ed. McGraw-Hill.