NetClinic: Interactive Visualization to Enhance Automated Fault Diagnosis in Enterprise Networks

Zhicheng Liu* Microsoft Research, Georgia Institute of Technology
Bongshin Lee† Microsoft Research
Srikanth Kandula‡ Microsoft Research
Ratul Mahajan§ Microsoft Research

ABSTRACT

Diagnosing faults in an operational computer network is a frustrating, time-consuming exercise. Despite advances, automatic diagnostic tools are far from perfect: they occasionally miss the true culprit and are mostly only good at narrowing down the search to a few potential culprits. This uncertainty and the inability to extract useful sense from tool output renders most tools not usable to administrators. To bridge this gap, we present NetClinic, a visual analytics system that couples interactive visualization with an automated diagnostic tool for enterprise networks. It enables administrators to verify the output of the automatic analysis at different levels of detail and to move seamlessly across levels while retaining appropriate context. A qualitative user study shows that NetClinic users can accurately identify the culprit, even when it is not present in the suggestions made by the automated component. We also find that supporting a variety of sensemaking strategies is a key to the success of systems that enhance automated diagnosis.

Keywords: Sensemaking, Semantic Graph Layout, Visual Analytics, Network Diagnosis, Information Visualization

Index Terms: H.5.m [Information interfaces and presentation (e.g., HCI)]: Miscellaneous;

1 INTRODUCTION

Network diagnosis is the task of finding the root cause of observed faults. The complexity of modern computer networks makes diagnosis a difficult, frustrating, and time-consuming exercise. To help system administrators, much research and commercial work has gone into building automatic diagnostic tools [1, 3, 6, 9, 17].

Despite recent advances, due to the difficulty of the diagnosis problem, automated tools do not always provide an accurate diagnosis. They occasionally miss the true culprit and commonly can only reduce the search space to a small number of likely culprits. To complete the diagnostic task, system administrators need to make sense of the output (i.e., probable causes) of the automated tools and, when the output is incorrect, they have to manually identify the correct culprit. The large amount of data and the sophistication of the required analysis make both these manual tasks challenging. The administrators need to examine the underlying raw data (e.g., various performance indicators of individual applications) for anomalies and correlations in addition to the diagnostic engine's analysis. As we explain in §2, such analysis typically operates at multiple levels of detail, with higher levels building on the results of lower levels.

As a "deliberate effort to understand events" for the purpose of detecting problems and explaining anomalies [16], network fault diagnosis is a sensemaking activity. It is argued that interactive visualization combined with computational data analysis more effectively supports the sensemaking process [27]. A few existing systems integrate data analysis with interactive visualizations [21, 26]. However, they employ computational techniques, such as entity extraction, clustering and dimensionality reduction, to reduce data complexity and provide better inputs for visualization. Users still have to perform exploratory analysis over the visualization.

In a departure from prior work, this paper explores the complementary relationship between visualization and a sophisticated computational analysis. In particular, we ask how useful visualization would be when an automated reasoning engine that solves the sensemaking problem is available? How should we design such a visual analytics system, and in what ways does it shape human sensemaking strategies.

We address these questions in the context of diagnosing faults in an enterprise network, which is a particularly complex type of computer network. By coupling interactive visualization with probabilistic *multi-level* diagnosis [13], we show that users are better able to use uncertain automated output and identify true culprits.

Our main contributions are:

1. Designing NetClinic, a novel visualization of directed graphs integrated with a *multi-level* automated analytic reasoning engine for network fault diagnosis. Its design is based on a semantic graph layout that lets users verify the correctness (or incorrectness) of the analysis at any level and move seamlessly across levels while retaining appropriate context.
2. Presenting results of a user study which shows that users use different sensemaking strategies and that the strategy of an individual evolves with experience. The flexibility of NetClinic in supporting these strategies allowed users to complete the diagnostic tasks with a high success rate even though they had no knowledge of the underlying diagnostic algorithms.
3. Articulating the lessons from our experience in the broader context of domains that combine interactive visualization with sophisticated computational analysis. These lessons have implications for the design of the visualizations, the automatic analysis, and their integration.

2 FAULT DIAGNOSIS IN ENTERPRISE NETWORKS

Enterprise networks consist of machines interacting with each other. Users often experience faults as anomalies in application behavior. For example, an email client (e.g., Outlook) is unable to send email. The root cause may be any one of the network components that influence the client application directly or indirectly (e.g., router firewalls, lossy links, bad server configuration or another client overloading the server). Diagnosis is the task of using available information to identify the true culprit.

Diagnostic systems model the interaction between network components as a dependency graph [6, 13]. Figure 1 illustrates a subset of such a graph, with several types of network components. Direct impact is depicted as an edge betwen components. It shows, for example, that an email server (depicted as 'emailserver.exe') depends on its client applications; a client that sends too many emails

*e-mail: zliu6@gatech.edu
†e-mail:bongshin@microsoft.com
‡e-mail:srikanth@microsoft.com
§e-mail:ratul@microsoft.com

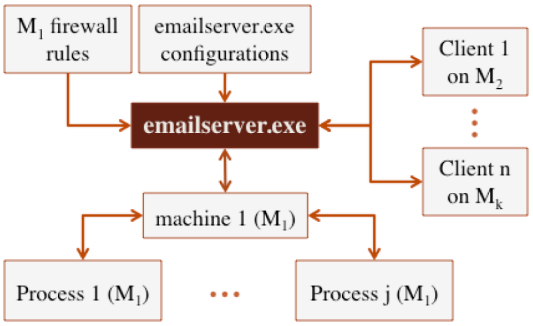

Figure 1: The dependencies of an email server application. Arrows indicate direction of potential impact.

may slow down the server. Similarly, changes in the server's configuration or on the machine it runs on directly impact its behavior. Indirect impact happens between components that are farther away in the graph. For example, other processes that run on the same machine can impact a process indirectly via changes in the machine. Thus, given a faulty network component, the culprit need not be an immediate neighbor. In fact, impact often propagates along a path of causal changes that can contain many components.

2.1 Diagnosis as Sensemaking

As a sensemaking activity, network diagnosis is a reciprocal process between data and mental structures. These explanatory mental structures, often called *frames* or *schemas*, are based on one's experience of the world, and are constructed from long-term memory to predict explanations and shape interpretations [16, 23, 25]. For example, given that none of the clients in the network could send email, system administrators may suspect some issues with the network connectivity. This frame leads them to look for data that either confirms or rejects it, e.g., incoming and outgoing packets at the machines. If they find that network traffic has been otherwise normal, they search for new data as anchors for a better frame. Diagnosis continues as the interplay between top-down and bottom-up processes, where they search for data that suggests a relevant frame, and the frame in turn determines what data to find for verification.

Abductive reasoning is believed to dominate the sensemaking process [18]. If the data matches a frame better than any other frame, that frame will be accepted as the likely explanation and sensemaking stops henceforth. Such reasoning, however, is a logical fallacy from a deductive point of view: given the premise that if the frame is true, the data is true, it does not hold that the data being true implies that the frame is true. In practice, sensemaking is more effective when people speculate about causes and rely on plausibility and reasonableness [18, 28].

A successful strategy for network diagnosis is to follow possible courses of events backwards from the observed anomalous effect [24]. Starting with the faulty component, diagnosticians may examine its state and look for directly impacting components that may cause the observed problem. Determining the state of a particular component is often non-trivial and requires examining a large information space. For example, the state of an application manifests in various aspects ranging from resource consumptions to error codes of its interactions. Further, tracing backwards from the component that is experiencing the fault involves repeatedly branching-off to observe other possible candidates. As a result, the search space quickly becomes too big to handle manually.

2.2 Automating Diagnosis

Due to this prohibitive search space, the development of automated tools to assist with diagnosis has been a subject for much research activity [5, 6, 9, 17]. These tools tend to use the backwards tracing strategy outlined above. Starting from *raw information*, they derive logical assumptions that are used for successively higher level reasoning until the conclusions are reached. The raw information for a network component is captured using one or more variables (also called *performance counters*) that describe its behavior. Different components have different variables. Examples include CPU usage, memory usage, and the amount of outgoing network traffic. The reasoning process can be divided across four semantic levels of detail with diagnostic systems differing in the algorithms they bring to bear at each level.

- *Variable Level*: At the lowest level, the diagnostic system determines which variables indicate abnormal behavior, often based on how different statistically the current values are to historical values.
- *Component Level*: Based on the analysis of individual variables of a component, the diagnostic system determines if the component as a whole is abnormal.
- *Edge Level*: Given two components in the dependency graph with an edge connecting them, the diagnostic system computes the edge weight, i.e., the likelihood of the source component *actually* impacting the target component. This computation is based on the state of both components.
- *Network Level*: Given a faulty component, a search in the entire network is conducted to find likely culprits, which are connected to the faulty component through a series of high weight edges. Based on these path level weights, the culprits are ranked from most likely to the least likely in the output.

Due to the complexity of the problem, unless a narrow set of faults are targeted, no diagnostic system can always provide an accurate diagnosis. Inaccuracies occur at all levels. At the variable level, for instance, if abnormality is statistically determined, a variable that behaves *better* than before may be deemed abnormal. As inaccuracies propagate to higher levels, they may be amplified when being combining with other inaccuracies. NetMedic [13], a recent automated diagnostic system that we build on in this work, has about 80% accuracy of identifying the true culprit.

3 NetClinic

The possibility of inaccurate diagnosis, even if it happens only occasionally, requires human users to verify the correctness of automatic suggestions and to identify the correct culprit when the suggestions are incorrect. Simply showing the output of the automated reasoning system without a human understandable version of the underpinning analysis limits the value of these tools. Rather we would like to show the basis of that output (e.g., network topology, component state) in a form that lets users explore data and make sensible decisions [15].

To this end, we advocate combining automatic diagnostic systems with visualization for effective sensemaking exploration. There are several design considerations based on our earlier discussions of human sensemaking and machine reasoning. First, the outputs of automated engines, albeit not always accurate, should be used whenever possible as they serve as useful frames to guide the sensemaking process. Visualization can show the analysis outputs for all the levels so that users can start sensemaking at any level of abstraction. Secondly, the abductive and reciprocal nature of human sensemaking should be supported. The visualization thus must not impose constraints on navigating across levels of abstraction. Top-down exploration lets users quickly verify the output of automated analysis at each higher level by looking at the information at the lower level. For instance, users can verify the edge level analysis, by looking at the states of the components on either end of the edge. Bottom-up exploration, going from lower to higher levels, lets users form and evaluate their own hypothesis. For instance, users can estimate which neighbor impacts a component the most by looking

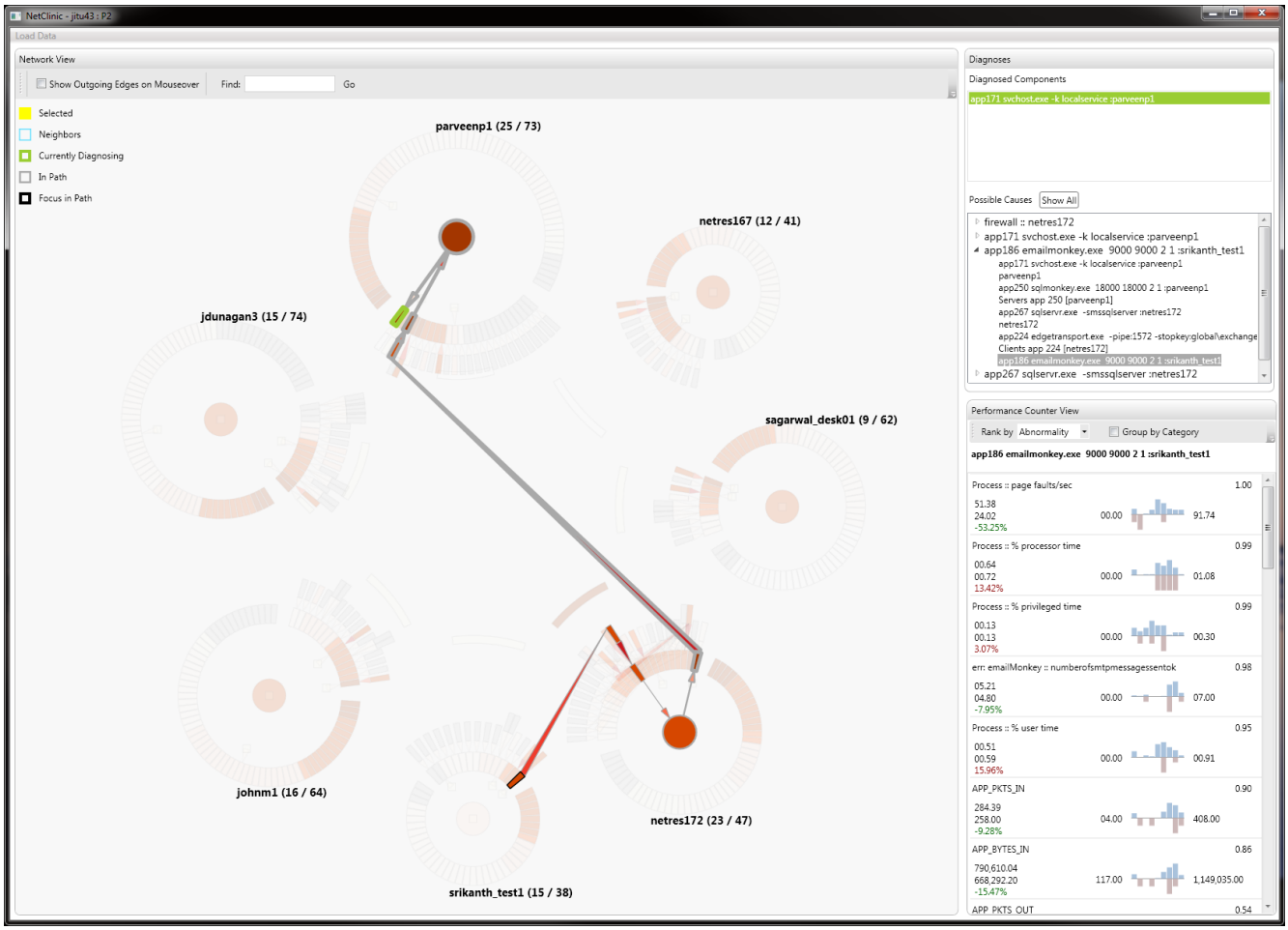

Figure 2: NetClinic consists of three parts: 1) A Network View on the left visualizes the network at component, edge and network levels; 2) the Diagnosis View on the upper right presents suggested diagnoses at the network level; and 3) the Performance Counter View on the lower right displays variable level data. The graph shows a network with 7 machines; one cause→fault path is highlighted.

at the states of these neighbors. Thirdly, raw information must be available for verifying machine generated outputs and supporting users' self-exploration.

We designed NetClinic (Figure 2), a visual analytics system for network fault diagnosis based on these considerations. The NetClinic interface consists of three areas: the main Network View on the left visualizes the network components and presents the information at *component*, *edge* and *network* levels; the Diagnosis View on the upper right shows NetMedic's diagnosis results at the *network* level; and the Performance Counter View on the lower right displays *variable* level and *raw* information about various performance counters associated with a component.

While we use NetMedic as the underlying analytic engine, NetClinic can be generalized to other diagnostic engines for enterprise networks because they share a common overall framework. We believe that its underlying techniques apply to other types of computer networks (e.g., Internet service providers) as well; enterprise networks represent the more complex setting as they tend to have a higher density of dependency edges.

3.1 Semantic-based Graph Layout

Diagnosis across even a handful of machines can involve hundreds of components, which makes the layout task challenging. To display a large number of components in a manner that users can relate to easily, we appeal to the semantics of the components and their relationships. For users, a machine is the most common grouping unit for network components. Applications belong to the machines that they run on, and communication between applications depends on the communication infrastructure between machines. We thus use a machine-oriented metaphor as the basis for our layout.

Visualizing links is a concern that is not independent from the graph layout. Due to the rich interaction between network components, the number of links is high – three times as many links as the number of network components in the examined enterprise network [13]. Showing all the links clutters the view due to edge crossings and occlusions. While edge bundling reduces clutter, it takes away attention from edges that represent high abnormality. Hence, rather than show every link all the time, NetClinic dynamically shows only links relevant to current user explorations. Further, it provides an easy interface for the users to customize the edge view (e.g., turn on/off different logical groups of edges).

Figure 2 shows a global view of the layout. Network components are grouped into machine clusters. Figure 3 shows a machine cluster in more detail. The *circular node* at the center represents the machine, and *segments* on a ring around the machine denote applications running on this machine. The sizes of the segments are normalized for visibility (e.g., in Figure 2, the machine at the top, *parveenp1*, has the most applications and hence the largest ring). Square boxes represent configurations: the machine configuration is placed at the center of the machine's circle, application configurations are placed adjacent to the application's segment on the ring (Figure 3). Machine clusters are grouped into a circular lay-

out (Figure 2). Applications with connections to applications on other machines are oriented to face the center of the layout, and a visual gap separates them from local applications.

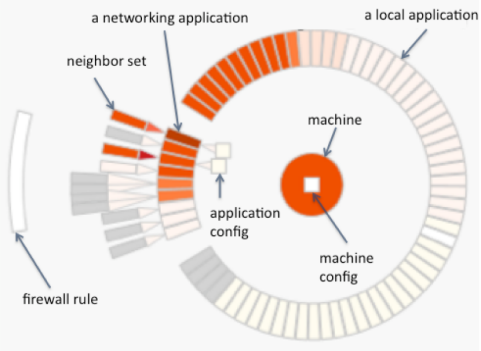

Figure 3: A machine cluster depicting the layout of various components. Colors represent the abnormality values of the components.

In addition to components that correspond to real network entities, diagnostic systems have some logical components. NetMedic uses a *neighbor set* to represent the interaction of an application with its peers. Neighbor sets of networking applications are laid out as segments along an additional outer arc so that when edges are shown, they will cross through the center of the circular layout and avoid occluding potentially relevant nodes.

Finally, firewall rules at each machine are represented as *arcs* that satellite the machine clusters. Since firewall rules impact every communication to and from the machine, these nodes have a high degree and we place them close to the center of the circular layout. Some diagnostic systems may have additional component types (e.g., routers, which NetMedic does not diagnose). We can extend the white-space in the center of our layout to include them.

For this design, the layout algorithm computes two important parameters: the normalized application segment length and the coordinates of the center of each cluster. With these two parameters, we can calculate the distance from the application segments to the cluster center for each machine. Algorithm 1 shows how we compute the parameters. Overlaps between components happen rarely and we found them to not severely impact user exploration. Hence, we focus on efficiency; this algorithm has linear time complexity.

Algorithm 1 Semantic Layout

Define N_i : number of applications on machine M_i
Define $C(X_i, Y_i)$: Cartesian coordinates of the center of M_i
Define $P(\theta, r)$: Polar coordinates with angle θ and radius r
$\theta = -\pi$;
for each machine M_i **do**
  $\alpha = 2\pi * N_i / \sum_i N_i$
  $Length_{segment}$ = $\pi / Max(N_i)$*canvas_width*sin(π / N_i)
  $Radius_{cluster}$ = canvas_width/2 - $Length_{segment} * N_i / 2\pi$
  $C(X_i, Y_i)$ = ConvertToCartesian(P($\theta + \alpha/2, Radius_{cluster}$))
  $C(X_i, Y_i)$.offset(center of the canvas)
  θ += α
end for

A remaining challenge is scaling the visualization to a large number of machines. Some enterprise networks may have hundreds of machines. Our experience in large enterprise networks [6, 13] indicates that most machines are redundant for the purpose of diagnosing any particular fault. Any diagnostic task has only a few machines pertinent to the fault (the server(s), some clients(s) experiencing the problem and the likely culprits). Since machine-level diagnosis is an easier problem [6], automated tools can discover the set of machines pertinent to a fault. As future work, we are examining ways that allow users to add or remove machines interactively and support semantic zooming that scales the amount of visual details per machine. On a 21 inch monitor, we find that the detailed layout scales easily to ten machines, which was sufficient for all the examined faults in a substantial dataset.

3.2 Contextual Exploration Across Analysis Levels

Given a set of machines under investigation, the semantic graph layout in NetClinic stays stable and supports both top-down and bottom-up exploration across multiple levels of information.

3.2.1 Top-down Exploration

A common starting point for diagnostics is the topmost *network level*. At this level, users see the diagnosis results computed by NetMedic as an impact path, that is, a path from the culprit to the component being diagnosed. This is achieved through the coordination between the Network View and the Diagnosis View (see Figure 1 for definitions). When users double-click on a component, NetClinic shows diagnostic results in the Diagnosis View. To allow diagnosing multiple components in one session, this view has a list at the top that tracks the components being diagnosed; the active component (*victim*) is highlighted in green (Figure 2). Another list at the bottom shows the top 5 likely culprits of the active victim. By default, the first culprit is selected and the path from the culprit to the victim is highlighted in the Network View; the components on this path stay in full opacity, while the others fade into the background. This path visualization immediately tells users the nature of the hypothesis, without worrying about the exact components involved. For example, in Figure 2, a culprit application on *srikanth_test1* is impacting the victim application on *parveenp1* through two applications on *netres172* and one on *parveenp1*.

To analyze a path in detail (e.g., to verify its accuracy), users can access the relevant *edge* and *component* level information. Direction is shown as an arrowhead atop a straight line for edges inside a machine cluster and a tapered representation for edges across machines. Edge color encodes the weight value computed by NetMedic's *edge level* analysis. The darker the red, the more likely the source impacts the target. Mouse-over an edge brings up a tooltip that shows the names of the two surrounding components and the computed weight value. Similarly, the color of a component encodes the abnormality computed by NetMedic's *component level* analysis. When the values are missing for certain components, which may happen because the application is no longer running or in rare cases due to data loss, we color the component nodes gray. Mouse-over a component shows a tooltip with the components' name and the computed abnormality value.

Users can drill down into the *variable level* and *raw* information to verify and understand what is abnormal at a component or why an edge has a high impact by using the Performance Counter View. Each row represents one observed variable for the component and rows are sorted by the abnormality values of the variables. A row has the name of the variable on the left, its abnormality value (the result of *variable level* analysis) on the right and *raw* information (e.g., performance counter values) in the middle. Raw information from two periods –the current diagnosis period and a historical no-fault period– for comparison, is plotted as a histogram. The horizontal axis represents the range of the values, divided into a number of bins, and both minimum and maximum values are shown. The vertical bars represent the frequency of values in each bin. We use different colors, blue for historical data and brown for the current data, for easy visual comparison. NetClinic also shows averages of the historical and current values and the percentage change in average on the left of the histogram. Together, the numerical values and histogram, let users quickly determine if the deviation between historical and current values is semantically meaningful.

For example, Figure 2 shows the variable level and raw information about the application emailmonkey.exe in the Performance Counter View. Note the second row where NetMedic assigns a high abnormality value (0.99 out of 1) to the variable `Process :: % processor time` through its statistical abnormality detector, yet the histogram shows little discrepancy between the historical and current distributions with the maximum of the raw data at 1.08%. Semantically, this is too low to be abnormal for CPU utilization and can be safely ignored. In this way, users can quickly sort through the multitude of counters, determine which variables are indeed abnormal and which neighbors are worthy of blame.

3.2.2 Bottom-up Exploration

Based on the description above, we can see how the visualization of raw information lets users reach variable level hypotheses, and how variable level information lets them reach component level hypotheses. To go from component to edge level, at any time, users can mouse-over a component to highlight both the component and its adjacent neighbors with directed edges joining them. Clicking on a component marks it in yellow and highlights its adjacent neighbors (Figure 4). Users can hide outgoing edges from the component because incoming edges are more important for diagnostic sensemaking when backtracking from the effect.

In this way, users can explore the neighbors of any component that are not included in the diagnosis paths. By examining the abnormalities and performance counters of adjacent components, users can reach edge level hypotheses (i.e., if impact really flows along an edge). For example, a causal impact is not likely if the source application is sending too little data and the target application is consuming too much memory, even if both components are independently abnormal. By tracing a chain of high impact edges, users can reach a network level hypothesis.

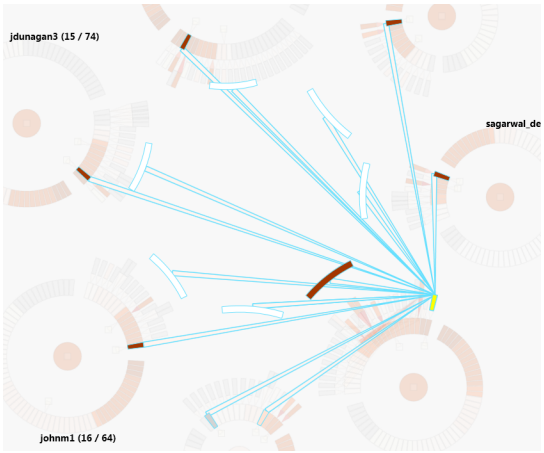

Figure 4: Marking the client set of a server application shows components that can potentially impact it, i.e., its clients and firewall configuration along network paths.

3.3 View Coordination and Common Path Encoding

To preserve context as users explore, NetClinic supports a variety of features that let them keep focus and save cognitive work. First, substantial coordination between the views minimizes mouse clicks. Selecting a cause in the Diagnosis View highlights the corresponding cause-effect path in the Network View. The cause can be expanded to list all the components on this path; traversing the list highlights the component receiving focus with a thick black border in the Network View and the Performance Counter View updates to show counter information about this component. Second, we observe that the diagnosis paths for the top five suggested causes often share components and edges. For example, the path of impact for multiple diagnoses may pass through a machine though the eventual culprits are different applications on that machine. Knowing that a path is common, users can save work by not re-verifying the common part of the path. Hence, NetClinic provides visual cues about the common path by using a thicker border with thickness corresponding to the frequency of appearance across the top causes. Third, NetClinic also enables users to see the union of all top 5 paths simultaneously by providing a "Show All" button in the Diagnosis View. Finally, when a component appears in multiple diagnosis paths, if users determine that the component could not possibly impact the effect while examining one path, they can mark the component as uninteresting. Such a component is given a blur effect so that users can quickly rule out other paths that this component appears on.

4 Evaluation

We conducted a qualitative user study to examine the usability of NetClinic and to understand users' diagnostic strategies with NetClinic. To our knowledge, NetClinic is the first system that couples interactive visualizations with an automated reasoning engine for fault diagnosis. The state-of-the-art visualizations (e.g., PerfMon [4] in Windows™) are rather primitive; they allow customizable displays of raw information but do not support analytics. Due to the lack of equivalent systems, we decided not to run a comparative study but to focus on how well NetClinic achieves the intended tight coupling between visualizations and analytics.

We used data collected by NetMedic during its one-month deployment in an enterprise network [13]. A diverse set of faults that is based on an earlier study of problems reported by network operators [13] was injected into the network. Since the faults are injected, the ground truth information on culprits is known and is used to evaluate users' diagnostic conclusions.

4.1 Methodology

We recruited eleven participants (excluding one pilot), ten of whom were male. They were graduate students except for one system engineer. All were working on computer networks or operating systems. They were not familiar with the diagnostic algorithms used by NetMedic and had never seen NetClinic before.

We began the session with a broad introduction on network diagnosis without specifics of the algorithms used by NetMedic. We then gave a tutorial on NetClinic and explained its features. Together, these lasted 20 minutes. The participants performed two training tasks with guidance, each with a small network consisting of four machines and a graph of 243 nodes and 683 links. In each task, the goal was to identify the real culprit for a specific victim component. The real culprit was in the top five causes suggested by NetMedic for one task and not in the other. It took up to an hour to finish the training tasks.

The participants then performed three independent test tasks, with the goal of identifying the culprit for three faults. They were told that the real culprit might or might not be in the top five causes suggested by NetMedic. To counter-balance the effect of automated diagnosis accuracy, the real culprit was in the top five causes for two (out of the three) tasks for half of the participants and one task for the other half. The network used in the test tasks consists of 7 machines, 682 components, and 2045 edges. Table 1 summarizes the tasks used in the study. The first two tasks are used for training, the remaining three are used in the actual tests.

The participants were asked to think aloud and we observed and video-recorded their sensemaking processes. Since domain knowledge is crucial to successfully complete the tasks, the participants were allowed to ask specific questions on the semantics of the components and performance counters. They were asked to prepare a

Symptom of Fault	Cause
The email client on a machine is experiencing some errors	The client's configuration is broken
Some SQL clients are experiencing poor performance	Another client is overloading the server
An email client cant get up-to-date data from server	The remote drive is dismounted
Some users were unable to access a specific feature of a Web-based application	The firewall along the path was blocking https traffic
Some clients cannot connect to the database server	A port used by the problematic clients had been blocked by a change in firewall rules on the server machine

Table 1: Symptoms and causes of faults used in user study

short reasoning statement for each fault they diagnosed. At the end of the session, the participants filled out a satisfaction questionnaire and provided feedback in a semi-structured interview.

4.2 Results

All the participants completed the three test tasks in about one hour. Eight out of the eleven participants correctly identified the culprits for all three faults. Overall, the culprits were correctly identified in 29 out of 33 tasks (88%). Recall that the real culprit was present in the suggestions from NetMedic only 50% of the time (2/3 tasks for half the users, 1/3 for the rest) and even when present, its order in the list of suggestions was randomized. We find this result encouraging, given that network diagnosis is a specialized task and that participants knew little about the algorithms used by NetMedic. This indicates that effective visualizations free administrators from needing intimate familiarity with the complex analysis done by the diagnostic engine while being robust to the uncertainty in the output of the engines.

4.2.1 Sensemaking Strategies Using NetClinic

In this section we describe the sensemaking strategies used by the participants to diagnose faults. In general, we observed that the process of iterating between frames and data still applied when the participants used NetClinic. However, since most participants had no prior system admin experience, only some could generate anticipatory frames from the fault description. NetMedic's diagnoses helped here: all participants used them to postulate frames. Verifying these diagnoses was a large part of the sensemaking process.

We expected the participants to adopt a "least-effort" strategy, where they would first look at each of the top 5 diagnoses. Only when the data did not fit into any of these suggestions did we expect them to explore on their own. Some participants did adopt this strategy as evident in this comment:

> "*I would use [the diagnoses] as the first pass, if they have anything that looks promising I will pay more attention to it, but I wouldn't completely just believe in the results given by the diagnosis*." (P3)

To our surprise, however, a majority of the participants did not adopt this strategy. Even while verifying a suggested diagnosis, participants would sometimes go off-track to examine interesting nearby components (based on their color, thickness of edges leading to them) along the paths. After looking at just one or two of the diagnoses, they would go back and forth between top-down and bottom-up exploration. Effective support for exploration was more important than displaying the suggested diagnoses well.

> "*I was trying to first find the problem myself. I started with the first [diagnosis], I wanna make sure this is definitely not the cause or it's definitely the cause. That's why I spent the most time on the first option, and doing that exploration I had a good feeling of what the actual cause is, so when I went to the second and the third and the other ones, I already knew what I was looking for, so I didn't spend much time on that*." (P1)

We also found that the same participant would vary strategy over time due to the increasing familiarity with NetClinic or based on their experience diagnosing the earlier tasks and the specifics of the fault they were diagnosing. Towards the end of the session, some participants generated frames and began exploration, from any component at any level, independent of the diagnoses. If the initial frame was rejected, they would return to use the suggested diagnoses as anchors for new frames and look for what they missed:

> "*The first thing I did was to look at how abnormal the problem node really is, and then given my prior experience the first thing I also check was, was there any configuration change on local machine or on the node itself. These are something that can easily go wrong. After that I let the diagnoses guide me, though what I did very often was deviating from them, going deeper in one direction that I think might be worthwhile. Though I always try to check these possible causes just not to overlook something*." (P10)

One participant did not make use of the diagnoses at all for one task. This complete bottom-up exploration strategy was less common. He did not like to see the diagnosis results, partly because he was not sure how to remove the diagnosis path after examining it:

> "*If I mouse over I see a whole lot of stuff and I felt that's better. Also in one of the [previous tasks] the top 5 did not seem to have the answer and I said 'forget it man' because once I get a hang of the tool I began to feel like 'ok, I can actually walk through the path myself'*." (P5)

In summary, we find that the participants use a rich mix of strategies. These strategies ranged from complete top-down exploration (for verifying results), switching between top-down and bottom-up explorations (for gaining confidence by looking at neighboring components), starting exploration at intermediate levels (for confirming specific hypothesis), and complete bottom-up exploration (for identifying the culprit without the help of the diagnostic results). NetClinic effectively supported all these strategies.

4.2.2 User Satisfaction and Usability

The satisfaction questionnaire consists of 11 statements with ratings on a 1-7 Likert scale, 1 being "strongly disagree" and 7 being "strongly agree." In general, the participants rated NetClinic favorably. They liked NetClinic (avg. 5.9) and the Network View in particular (avg. 6.3). Features of the Network View, the focus+context highlighting (avg. 6.6), the color encoding (avg. 6.4) and the semantic layout (avg. 6) were also considered useful. While most participants find the Performance Counter View helpful (avg. 6), their reactions to the histograms in the view were mixed. Some found them intuitive and could quickly run through the counters and identify interesting ones. Others did not really understand them and relied on the numerical values. We think increased familiarity with NetClinic by more training might help in the latter case.

Two major concerns were related to ease of learning (avg. 5.7 with a minimum of 3) and mistake recovery (avg. 5.4). Not surprisingly, some participants thought it was difficult to learn NetClinic. Participants have to familiarize themselves with not only the interface but also the diagnostic reasoning related to computer networks. These two aspects can be inextricably linked especially for novice users. Learning hence is non-trivial, but we believe that the increasing ability to diagnose faults faster than otherwise possible will offset learning cost. Though we lack statistical evidence, we found users speed up over the duration of their session.

Many participants noted that when they accidentally cleared their selection by clicking on the background, they could not recover from the mistake. A sudden change of the context can be detrimental; an undo feature would have helped here. Some participants

wanted more visual cues. For example, they could not distinguish clients from servers and had to resort to reading the tooltips. In addition, a few participants were confused by neighbor sets (client sets and server sets) as these virtual components did not match their mental models. This raises the question of how to represent concepts that are important for computational analysis but are not intuitive to users. We believe this is an interesting problem and it is worthwhile to explore this further.

It is well known that human working memory is limited. One participant remarked that he often had to look at the relevant data again to reconfirm previous findings. Enabling users to offload information as external representations will be useful, perhaps with a separate shoebox or an integrated *edit* option.

4.3 Study Limitations

Due to the resource constraint, we did not to recruit system administrators as participants. While our participants lack practical administrative experience, they have adequate knowledge about computer networks. Considering them as novice system administrators, we believe they still provide insights into the usability and usefulness of NetClinic. Further evaluation with real network administrators (especially a longitudinal study of long term use of NetClinic) would shed more light on the benefits of NetClinic.

In this study, we focused on how well NetClinic achieves the tight coupling between the interactive visualization and an automatic diagnostic engine. The coupling of visualization with an analytic engine implies that we could compare NetClinic with different visualizations (e.g., tightly-coupled multiple tables) coupled with NetMedic or with visualizations that do not have an underlying analytic engine. Each of such comparisons would generate more insights on various aspects of the design.

5 Discussion

In this section, we articulate the key lessons learned from our work of combining visualization with an automatic diagnostic engine. Some of them re-confirm findings of prior work in a significantly more complex domain. We point out the implications of these lessons for the design of the visualization, the analytic techniques, and the combination of the two. We believe that these lessons apply broadly to other domains where visualizations can enhance complex computational analysis.

First, we find that *the combination of visualization and the automatic analysis makes diagnosis systematic yet flexible* [21]. NetClinic is systematic in that the multiple suggestions and the data analysis at each level of detail is independent of human bias towards interpreting data into existing frames and expectations. For example, P10 used the analysis to make sure nothing important was overlooked. Automatic analysis alone however is not flexible. NetClinic is flexible in that the visualization supports an opportunistic mix of top-down and bottom-up exploration by letting users easily move across different levels of detail. As a result, participants demonstrated a variety of strategies during fault diagnosis, and despite unfamiliarity with analysis algorithms, achieved a high success rate at finding the culprits.

Secondly, *automated analysis reduces the cost of assessing and selecting data for detailed attention*. The use of pre-attentive color and pattern coding to represent analysis results, which were driven by the automated analysis engine, served as important cues to let users quickly determine which component, edge or performance counter was worth examining. An important caveat is worth noting. Mistakes in the automatic analysis, be they false negatives (e.g., abnormal components are deemed normal) or false positives impact sensemaking. For example, the color-coding can hide the true culprit or force users to examine unnecessary components. *This asymmetry in the impact of errors has important implications for trade-offs in the design of automatic analysis*. For one, the analysis tool should favor false positives over false negatives to trade-off more user work for correctness. Exposing confidence data along with a decision would also be useful.

Thirdly, in the visualization, *maintaining a consistent context that users can relate to is important for them to track the evidence and hypotheses being examined*. In NetClinic, this is enabled by the semantic and explicit representation of the graph structure and the focus+context highlighting of components, edges, and paths on top of a stable layout. The participants extensively explored the graph incrementally and commented that it was great to be able to follow the dependence relations. Spreading out the machines on a semantic circular layout enhanced visibility and clarity. NetClinic's stable layout serves as external memory and makes it easier to remember previous findings. As mentioned earlier, the participants did not use the common path encoding and the "mark as uninteresting" functionality much. We believe that this was partly because the participant's visuo-spatial memory was sufficient to recognize components appearing in multiple diagnoses. Visuo-spatial memory has its limits, however; when the analysis suggested culprits that were close to each other in the layout, the participants had to read the labels and tooltips more closely.

Lastly, *tight integration of visualization and automatic analysis simplifies the sensemaking process*. Currently, the most common way of integrating automated analysis and visualization is to use computational techniques to reduce data complexity and create better visualization-data mapping; automated analysis and visualization often constitute disjointed processes. Because humans often organize their thinking in terms of varying levels of abstraction, tighter integration through level-specific coupling between automatic analysis and visualization lowers the cognitive overhead of sensemaking. Exposing information at different levels of abstraction is a commonly used technique included in design guidelines for information visualization [7]; on the other hand, data abstraction and hierarchical representations are often used in intelligent system design. Although the levels of abstraction in human sensemaking and automated reasoning may not always match, we speculate that they can be reconciled in many domains.

6 Related Work

Much existing work uses visualizations to enable real-time security monitoring and intrusion detection in computer networks (e.g., [10, 11, 19]). These systems focus on being at or near real-time. Fault diagnosis is different from monitoring and intrusion detection tasks. For example, to detect attacks, it is important to identify temporally related events such as connection initiations and failures [10]. Fault diagnosis is more concerned with component states rather than user activities and may require customized design. SCUBA [12], nCompass [2], and MTreeDX [20] among others do provide visualizations to help troubleshoot wireless mesh, enterprise and multicast networks. Most of these systems however only enable easy access to the raw data at the variable level without leveraging much analytics.

Although analytics has been absent in many of the computer network visualizations, some recent systems integrate data analysis techniques with more general network visualizations and study how users use such systems. Jigsaw visualizes connections between entities extracted from text documents [26]. Kang et al. [14] study how Jigsaw influences investigative analysis strategies by comparing it to traditional interfaces. SocialAction uses coordinated views and attribute ranking to support systematic yet flexible exploration of social networks [21]. Perer and Shneiderman conduct longitudinal case studies on data analysts' strategies in using SocialAction [22]. In these systems, the computational analyses are generally less sophisticated and do not directly provide hypotheses for the problems. D-Dupe [8] performs author name resolution and

uses a semantic layout to present analysis results. Resolving author names, however, is relatively simple and only a sub-task of a larger sensemaking process.

The novelty of NetClinic lies in the coupling between visualization and a sophisticated reasoning engine. First, since we treat analytics as an integrative component in the system design, the information to be visualized in NetClinic goes beyond raw information to include analytical outputs available at multiple levels of abstraction. Explicit design considerations about the flow of human sensemaking processes hence are necessary, which are typically absent in previous work. Secondly, these considerations entail design requirements on the network layout. For network fault diagnosis, it is useful not only to know which components an abnormal component is connected to, but also to have contextual information such as whether the neighbor is co-located on the same machine. Such semantic relationship information is crucial as it determines whether or not the neighbor can impact the component (e.g., high CPU usage of a neighbor is not a problem if the neighbor is on a different machine). NetClinic develops a new semantically meaningful layout, which as we showed above plays a key role in usability.

7 Conclusion and Future Work

We presented NetClinic, a visual analytics system that couples interactive visualization with an automated reasoning engine. We discussed the benefits and shortcomings of NetClinic based on the results and findings from a qualitative study. Drawing from our user study observations, we argue that introducing an automated reasoning engine simplifies the fault diagnosis task. To let users make the best use of such an analytic engine without sacrificing the flexibility of self-exploration, it is useful to expose information and support seamless exploration across all analysis levels.

The initial success of our system opens many avenues for future research. After refining NetClinic based on user feedback, we plan to deploy NetClinic in an enterprise. We are especially interested in the evolution of administrators' strategies over a longer term.

Another promising avenue that we have begun exploring is understanding how to support modifying the automatic analysis itself. For instance, if users find that the edge level analysis for a particular edge is incorrect, they can modify the weight of that edge using a slider control. Or when crucial data for a component goes missing, they could supply hypothetical constructs for analysis. The diagnostic engine can take these as input to re-compute a new set of diagnostic results. Such two-way human-machine interaction improves the capabilities of the diagnostic engine and further simplifies the fault diagnosis task. We learned that using such features however poses two challenges. First, it places a high expertise requirement on the users; meaningfully modifying edge weights requires a deep understanding of both the domain as well as the diagnostic algorithms. Second, not all diagnostic engines support incremental re-computation and hence the visualization may become less interactive.

Acknowledgements

We would like to thank Danyel Fisher for contributing to the early stages of this project. We also would like to thank George Robertson and John Stasko for critiquing earlier versions of this paper.

References

[1] Gteko, inc. http://www.gteko.com, Mar. 2009.

[2] nCompass for Enterprises. OPNET Technologies, Inc. http://www.opnet.com, Aug. 2009.

[3] OpenView, HP technologies inc. http://www.openview.hp.com, Aug. 2009.

[4] PerfMon. http://msdn.microsoft.com/en-us/library/aa645516%28VS.71%29.aspx, Aug. 2009.

[5] S. Alexander, S. Kliger, E. Mozes, Y. Yemini, and D. Ohsie. High speed and robust event correlation. *IEEE Communication Magazine*, 5:433–450, 1996.

[6] P. Bahl, R. Chandra, A. Greenberg, S. Kandula, D. A. Maltz, and M. Zhang. Towards highly reliable enterprise network services via inference of multi-level dependencies. *SIGCOMM Comput. Commun. Rev.*, 37(4):13–24, 2007.

[7] M. Q. Baldonado, A. Woodruff, and A. Kuchinsky. Guidelines for using multiple views in information visualization. In *Proc. AVI*, pages 110–119, 2000.

[8] M. Bilgic, L. Licamele, L. Getoor, and B. Shneiderman. D-dupe: An interactive tool for entity resolution in social networks. In *Proc. IEEE VAST*, pages 43–50, 2006.

[9] M. Chen, E. Kiciman, E. Fratkin, A. Fox, and E. Brewer. Pinpoint: problem determination in large, dynamic internet services. In *DSN*, pages 595–604, 2002.

[10] R. F. Erbacher, K. L. Walker, and D. A. Frincke. Intrusion and misuse detection in large-scale systems. *IEEE CG&A*, pages 38–48, 2002.

[11] J. R. Goodall, W. G. Lutters, P. Rheingans, and A. Komlodi. Focusing on context in network traffic analysis. *IEEE CG&A*, 26(2):72–80, 2006.

[12] A. P. Jardosh, P. Suwannatat, T. Hollerer, E. M. Belding, and K. C. Almeroth. SCUBA: focus and context for real-time mesh network health diagnosis. *LNCS*, 4979:162–171, 2008.

[13] S. Kandula, R. Mahajan, P. Verkaik, S. Agarwal, J. Padhye, and P. Bahl. Detailed diagnosis in enterprise networks. *SIGCOMM Comput. Commun. Rev.*, 39(4):243–254, 2009.

[14] Y. Kang, C. Görg, and J. Stasko. Evaluating visual analytics systems for investigative analysis: Deriving design principles from a case study. In *IEEE VAST*, 2009.

[15] G. Klein, B. Moon, and R. Hoffman. Making sense of sensemaking 1: Alternative perspectives. *IEEE Intelligent Systems*, 21(4):70–73, 2006.

[16] G. Klein, J. K. Phillips, E. L. Rall, and D. A. Peluso. A data-frame theory of sensemaking. In *Expertise Out of Context: Proc. 6th Intl Conf. Naturalistic Decision Making*, 2006.

[17] R. R. Kompella, J. Yates, A. Greenberg, and A. C. Snoeren. IP fault localization via risk modeling. In *NSDI*, volume 2, pages 57–70. USENIX Association, 2005.

[18] C. G. Lundberg. Made sense and remembered sense: Sensemaking through abduction. *Journal of Economic Psychology*, 21(6):691–709, Dec. 2000.

[19] F. Mansmann, D. A. Keim, S. C. North, B. Rexroad, and D. Sheleheda. Visual analysis of network traffic for resource planning, interactive monitoring, and interpretation of security threats. *IEEE TVCG*, 13(6):1105–1112, 2007.

[20] J. Mulik, P. Conrad, B. Drake, S. Biswas, and M. Sendula. MTreeDx: a multicast network diagnosis tool. In *Integrated Network Management*, pages 105–108, 2003.

[21] A. Perer and B. Shneiderman. Balancing systematic and flexible exploration of social networks. *IEEE TVCG*, 12(5):693–700, 2006.

[22] A. Perer and B. Shneiderman. Integrating statistics and visualization: case studies of gaining clarity during exploratory data analysis. In *Proc. CHI*, pages 265–274, Florence, Italy, 2008. ACM.

[23] P. Pirolli and S. Card. The sensemaking process and leverage points for analyst technology as identified through cognitive task analysis. In *Proc. Intelligence Analysis*, pages 2–4, 2005.

[24] J. Rasmussen. Diagnostic reasoning in action. *IEEE Trans. on Systems, Man and Cybernetics*, 23(4):981–992, 1993.

[25] D. M. Russell, M. J. Stefik, P. Pirolli, and S. K. Card. The cost structure of sensemaking. In *Proc. INTERCHI*, pages 269–276, 1993.

[26] J. Stasko, C. Görg, Z. Liu, and K. Singhal. Jigsaw: Supporting investigative analysis through interactive visualization. *Information Visualization*, 7(2):118–132, 2008.

[27] J. J. Thomas and K. A. Cook. *Illuminating the Path: the Reseach and Development Agenda for Visual Analytics*. IEEE Computer Society, Los Alamitos, CA, 2005.

[28] K. E. Weick. *Sensemaking in organizations*. Sage, 1995.