ABSTRACT

The expansion of the Internet has led to the generation and collection of massive amounts of data, a fact which has both gains and costs associated with it. The data can be in the form of network packets, HTTP request data, error logs, or usage information, any of which may indicate trends that will inform business decisions or reveal problem areas that must be addressed. However the typical quantity of data collected renders visual analysis infeasible, a problem that Comcast operators experience regularly when trying to maintain their systems. Not only must they sift through all of the information collected, they must infer meaning from it. AnomVis uses a big data platform to parse system log files, and record trends in order to classify anomalous behavior in the system to display in a clean UI. The UI visualizations are created by employing cognitive metaphors, a human’s implicit associations between physical characteristics and the data they describe. Network operators can use AnomVis to help interpret the data they receive, identify problems in their systems, and determine actionable steps to resolve them. Through an iterative user study process we have honed the application and responded to qualitative and quantitative feedback to create a product that fulfills industry needs.

1. INTRODUCTION

Businesses with large technology infrastructures value internal network performance to maintain consistent operation of their system. By utilizing metrics such as the frequency of errors, number of network requests, and request round-trip time, companies can more easily detect potential inconsistencies or external attacks. However, identifying such inconsistencies in their systems is difficult due to the massive amounts of data produced and collected. Additionally, many companies only store a month of data at a time, ignoring trends developed over months and years. Thus there is an inefficiency in current standard workflows where we have identified a space to create value. A tool that studies metrics and notifies operators of anomalies could expedite this process.

Our solution combines analysis of historical trends and daily statistics, and data visualization with an emphasis on incorporating cognitive metaphors. We are focusing our efforts on creating a software application for Comcast to highlight potentially problematic metrics such as the number of requests to the server per user as well as the number of requests to each URL. The tool’s user interface aims to be easily understandable, allowing operators to glean information from it with little to no training. Our goal is to create a data visualization product that combines the powerful abilities of anomaly detection algorithms and big data software platforms with informed research on cognitive metaphors.

In this paper, we will first define some terms and give background about the field. Then we will discuss some related work in the various aspects of our application including anomaly detection, big data platforms and visualization tools for network data. Next, we will review how our solution works and explain some of our design decisions in the system model and implementation. We will also detail an evaluation of the effectiveness of our current UI design. Finally, we will go through the potential areas for expansion for the system along with a discussion on the ethical implications of our application.

2. BACKGROUND

Anomaly detection, or outlier detection, is the identification of events that deviate from the expected pattern in a data set[1]. In order to determine when events deviate from what is expected, a record of what has occurred in the past must be maintained. When new data is generated, statistical analysis can reveal how anomalous each point is relative to the historical record. In large datasets, where individual data points are easily overlooked, algorithmic analysis is a powerful tool to identify significant events and quantify their degree of abnormality.

Extracting meaning from this type of analysis is nearly infeasible without a graphical interface. Data visualization utilizes information graphics to communicate trends in large datasets to users. There are currently many tools that attempt this, however they typically do not incorporate anomaly detection to emphasize potential problems. Many of these products focus on displaying all of the data to the user and leave it as a task for the operator to sift through the large...
quantities of information and determine what points are anomalies. Additionally, these products can be overwhelming for the user and often do not provide enough context to determine a plan of action. For example, the graphic in Figure 1 resembles a typical dashboard, but provides no information about where the anomalies are.

![Sample Dashboard](image)

**Figure 1: Sample Dashboard**

Cognitive metaphor theory is particularly helpful when displaying data. A cognitive metaphor refers to the understanding of one idea in terms of another; an example of this is the understanding of quantity in terms of directionality in the phrase, “the prices are rising”. Taking this concept a step further, George Lakoff, in his book “Metaphors We Live By”, introduces metaphors that spatially organize a whole system of concepts with respect to one another[2]. The spatial organization of the system is not arbitrary; it is based off our physical and cultural experience. For example the cultural convention “More is up; Less is down” has a basis in the fact that adding more of a substance or physical objects to a container or pile makes the level go up.

This same concept can be applied to other cultural conventions such as “Important is bigger; Subtle is smaller”. By taking advantage of peoples’ implicit associations, visualizations can reduce the amount of information explicitly displayed in order to reduce clutter and overwhelmingness.

3. RELATED WORK

Large scale, networked systems have become ubiquitous and thus we need a platform capable of processing the quantity of data they produce. There are existing platforms for processing big data that can be used to implement anomaly detection. One such platform is Apache Spark[3]. It is an engine for large scale data processing and integrates with Hadoop. Apache Spark can be used in applications written in Java, Scala or Python. With its speed advantages over Hadoop MapReduce and its easy integration into any application, Apache Spark would provide us with a strong platform to execute anomaly detection over a large set of data.

Another existing platform is 0xdata[4], which implements a combination of algorithms to classify data and predict behavior; it is an open source prediction engine for Big Data Science. 0xdata combines several advanced algorithms, exploratory data analytics, and prediction engines to create a scalable platform for machine learning and predictive analytics on big data. Utilizing a robust tool such as 0xdata to manage anomaly detection guarantees our anomaly detection will match industry standards and allows us to focus on the data visualization aspect of our project.

Another existing platform is the Kale stack[5] developed by Etsy. Etsy monitors over a quarter of a million distinct metrics for its business and, therefore, needs a comprehensive anomaly detection solution. Skyline[6], Kale’s anomaly detection system, identifies metrics that appear problematic based off of an ensemble of different algorithms. An agreement amongst a majority of the algorithms are needed to label a metric as anomalous. Skyline comes with many standard algorithms, with the ability for developers to plug in additional algorithms they implement themselves. This abstraction provides an excellent basis to customize an anomaly detection system based off of the proper statistical tools for the given data. The Kale stack also comes with Oculus, the anomaly correlation component of the system. Once an anomaly is detected, Oculus will find all of the other metrics in the system which look similar so that a human user can compare the potentially anomalous data to past situations with similar trends.

For data visualization, there exists a wide range of software solutions. Products such as Tableau offer a large scope of solutions, allowing users to customize graphs, display selected data, and highlight data trends. They state “Anyone can analyze data with Tableau’s intuitive drag and drop products. No programming, just insight.”[7] This application could be further improved, however, by reducing the amount of insight required. This observation shows how the data analysis process could be even more automated, by highlighting potentially problematic data. Our application allows operators to minimize their response time and improve their efficiency.

Another data visualization tool, EtherApe[8], monitors network traffic and displays the activity graphically. Hosts and links change in size based on traffic, and protocols are color coded. These visualizations use cognitive metaphors, but the overall implementation is ineffective. Looking at Figure 2 the larger circles represent hosts with more traffic. Yet, the reason for the cone out to a circle from a single point and the purpose of the circles being positioned radially is unclear. The human brain has difficulty focusing on more than one item at once. EtherApe’s visualization would be much improved by having fewer panels open to distract users from the important information. The most important information is not given at first glance, it requires clicking in and opening a new pane to give bandwidth and other network traffic information. Finally, this visualization does not address any anomaly detection. The user is left to determine what data is considered anomalous rather than having EtherApe present the information. EtherApe acts as an example of how difficult it can be to convey large amounts of network data succinctly and illustrates the need for modular views in our application.

New Relic[9] is another analytical visualization software that aims to visualize and provide meaning to the performance metrics it tracks. The company’s flagship product is its Application Performance Monitoring (APM) service that follows the performance of critical transactions across the entire service-oriented application environment. It uses the metrics gathered to identify the performance impact of specific code segments within an application. While New Relic is an excellent service for the visualization metrics, it
Figure 2: EtherApe

is lacking in its anomaly detection and visualization features. The APM only presents the user with graphical representations of important metrics, but does not employ anomaly detection algorithms. The APM service tracks anomalous behavior through manual thresholds set by the user. Our product aims to replace this common industry practice.

We intend to improve upon the threshold detection technique that Comcast operators currently employ to monitor their network activity. The operators need to quickly view important metrics, assess performance, and determine how to address problems that the data may reveal, but none of the previously mentioned visualization products fully meets their needs. By incorporating anomaly detection algorithms with a software product that displays information in a clear way, we will improve Comcast’s ability to quickly and accurately identify anomalies in its system.

4. SYSTEM MODEL

AnomVis takes network data, runs it through data processing algorithms, and calculates metadata to detect anomalies within the dataset. These characteristics include historical trends, averages, and standard deviations. The conclusion from these analyses is visualized to the user in a clear way that reduces evaluation time. Overall, the product allows users to make faster, more informed decisions through a refined visualization of the data.

In order to accomplish this, AnomVis integrates data analysis and visualization as depicted in Figure 3. The application first needs to receive the data from an external source. The current application is run with prior datasets but eventually the implementation could be altered slightly to work with live data. Regardless of the source, the data is first cleaned and processed. The raw data is converted to the format expected by the front end while simultaneously being processed into a format the data processing component can use to calculate anomalous characteristics like historical trends and standard deviation.

Historical trends are stored per metric and URL, with each hour in a twenty-four hour day storing a record of its running average. The system maintains separate trends for weekdays and weekends to account for different usage patterns between the two. The standard deviation calculation works similarly, calculating the standard deviation from the historical trend for every day based on all of the previous days’ averages.

These properties are used to incorporate cognitive metaphor theory when presenting information to the user. Cognitive metaphors help operators intuit meaning more quickly by using inherent assumptions about color, size, position, etc. instead of distracting them with explicit labels. To further simplify the visualization and prevent users from being overwhelmed, the UI limits the information presented at one time. Only a small portion of the available data is displayed by default, and operators can gradually see and learn more as they interact with the application.

5. SYSTEM APPROACH

Within our system, the data is transformed from raw data to cleaned and processed data that can be injected into the front end. The lifecycle of a data point starts from the log file, where it is represented as a line containing 15 fields shown in Figure 4. The line is split into a list of fields, and the timestamp, URL, and response time or status codes are extracted depending on which metric is being analyzed. The metrics are grouped by URL, sorted by the number of requests in each group, and then filtered so that only URLs requested more than 30000 times that day are analyzed. This threshold was chosen to ensure that enough data for each URL was present, with the intuition that a URL requested infrequently would not be visually significant.

Figure 3: Architecture

Figure 4: Raw data point - 5:20:25, 10050 ms

For the response time metric, a K-Means model trains over all processing times logged during each two minute period of the day, with K equal to three unless there were fewer than three requests during that period. K-Means is “a simple and easy way to classify a given data set through a certain number of clusters (assume k clusters) fixed a priori.”[10] Each duration is then assigned to one of the clusters determined by the model; Figure 5 shows three additional points that would be clustered with the point shown in Figure 4. Finally, each cluster is condensed into the format shown in Figure 6, and added to the structure that eventually gets uploaded to the database.
As each cluster is calculated, the duration and count for that cluster are added to an accumulation of cluster values for the given hour. Once all clusters for an hour have been calculated, the hour average is added to the hour information in Figure 6. The durations and count for the hour are then stored in the back end trend file both as an incremental accumulation of all past values and in the “day list” as a record of all previous day values for a particular URL and hour as shown in Figure 7. These fields are used to calculate the overall trend average and standard deviation respectively. The calculated trend average and standard deviation values are stored in the processed data in order to maintain their values on the given day as more data is added to the trend file.

```
{  "hour_count": 27811.0,
   "hour": 17,
   "trend_avg": 51.250086247374009,
   "std_dev": 2.9600000310286568,
   "avg_dur": 247.93817315148022,
   "num_cluster": 1,
   "count": 280286.0,
   "avg_time": 12631,
   "cluster_id": 0,
   "avg_dur": 0.0000000000000000,
   "count": 5856.0,
   "avg_time": 12737,
   "cluster_id": 1,
   "avg_dur": 10808.0,
   "count": 14.0,
   "avg_time": 12808,
   "cluster_id": 2,
   "avg_dur": 7660.0,
}
```

**Figure 7: Trend record for 5 pm**

The data processing outlined for request durations is similar for status codes. Status codes, however, are not clustered due to the finite number of possible codes. Instead only the count, trend, and standard deviation are stored in a similar fashion for each status code, on a per-hour basis.

When the response times are visualized in the UI, the average response time for the hour and the historical trend for that hour are used to create the bullet chart shown in Figure 8. The bar represents the average response time for the hour on that day, and is red because the average for the hour is over two standard deviations away from the historical trend for that hour. The historical trend is represented by the black line, and the exact value is given when the user hovers over the line.

**Figure 8: Bullet Chart visualization of 5 pm**

In a more detailed view of the UI, shown in Figure 9, the clusters created in the processing are visualized in a scatter-plot of response times. The x-axis is the minute in the hour and the y-axis is the response time in milliseconds. The size of each point is proportional to the number of data points that are in the cluster. The particular point exemplified here is very anomalous, more than two standard deviations away from the trend. This is visually emphasized as it is far away from the blue trend line on the plot and is colored red.

**Figure 9: Scatter plot visualization of 5 pm**

Once that point is clicked, it is highlighted and information about that point is displayed as seen in Figure 10. This data gives the user a starting point to figure out exactly when the anomaly occurred and postulate sources of the outlying data point.

**Figure 10: Detailed cluster information**

In the UI, shown in Figure 11, the data for the status code metric is visualized in a bar graph. The colored bar represents the actual count of that level status code (e.g. 200) and the black bar represents the trend for that hour. The trend bar is black to act as a neutral comparison in
our color scheme. In the figure, the 200 level status code count is very typical and is colored green, whereas there is an anomalous number of 500 level status codes.

Figure 11: Status code visualization for 5 pm

Clicking on a bar, in Figure 12, the 500 level status codes are shown, providing more detail into which status codes are the outliers. The visual in this detailed view is the same type of bar graph comparing the actual frequency of a status code compared to the trend. In this case, it is clear that both 500 and 503 are more than two standard deviations away from the trend, causing them both to be red and considered anomalous.

Figure 12: Detailed visualization of 500's at 5 pm

This walkthrough demonstrates the process AnomVis goes through for each data point to be visualized.

6. SYSTEM IMPLEMENTATION

The application is implemented using Apache Spark for data processing, Amazon Web Services’ S3 buckets for data storage, Flask as the web framework and D3.js for visualization graphics. These tools provide the structure to analyze Comcast’s internal network data and ultimately display the results in a clean UI. The raw network data is in the form of anonymized HTTP request data and is stored by day in separate log files. For a month, the size of the network data is over 60 Gigabytes just for a single server.

The backend processes this data day by day as a daily task using Apache Spark, providing parallel processing functionality for the large quantity of data that the application must be able to handle.

Spark’s primary abstraction is its Resilient Distributed Dataset (RDD), into which the application first parses the textual log files. Spark then divides the processing of these RDD objects across the available cores of the machine and allows for the application of custom transformations to them. Relevant fields, such as URL, timestamp, status code, and request duration, are extracted and the lines in the input file are grouped by URL. The data for a request on a particular URL is then split up by the hour it was requested, and analysis is performed on hourly groups.

Analysis of request durations is achieved by using Spark’s built-in KMeans clustering algorithm to group the large number of requests into three clusters for every two minutes. This reduces the quantity of information sent to the web application while providing an average time, duration, and count to help categorize each cluster. To process each status code, a count of the number of occurrences is calculated for each hour. The hourly data for request response time and status codes are added to separate trend files that contain records from previous hourly analyses. The historical values are used to calculate overall trend averages and standard deviations for each URL per hour. After a day’s data for each URL has been refined, previous trends and standard deviations are added to the record for that day. Once all URLs for a day have been processed, the results are added to the trends for the month, stored separately from the daily trends, and the condensed monthly data file is updated.

The application runs on an Amazon EC2 instance to take advantage of the processing power and existing infrastructure. While testing this implementation with real time data is not a possibility due to security restrictions, the architecture was decided with live data in mind to make the switch as seamless as possible. After processing the data on EC2, it is uploaded to S3 buckets and stored there. S3 allows the application to easily and securely access the analyzed data. This tool is also very scalable, which is beneficial as more data is collected over months and years[11].

The D3.js library is used in order to effectively display the information processed through Apache Spark. D3 stands for Data Driven Documents, which stems from its focus on binding data to DOM elements and then manipulating those elements based on their data[12]. This is ideal for implementing cognitive metaphors, where the structure and orientation is defined in terms of the data being displayed.

The entire application displays information as a monthly overview, and then refines into a day-by-day breakdown, each divided up by hour. The landing page is a calendar heatmap of the data for a particular metric across all URLs during each month, giving the user an overview of each day. As depicted in Figure 13, each day is colored based on a scale from red to green, representing very anomalous to very typical data respectively. Anomalous data is determined based on how many standard deviations a data point is away from the trend. Red represents data points that are over two standard deviations away from the trend and green points are under half a standard deviation away. The color scheme is consistent throughout the entire application.

Furthermore, clicking on a specific day provides an in depth view of the data points for a specific metric. This
day view gives an overview of each hour on a color coded slider representing the anomalousness of each hour. The slider allows a user to select hours and see averaged data for a specific hour and the typical trend for that hour. Each hour is also broken down to more detail based on the metric. The two metrics that we chose initially are frequency of status codes and response times for each request. See Figures 14 and 15 for these views.

In the day view of the status codes metric, each hour shows a horizontal stacked graph that displays the count of each status code level from 200 to 500 in each hour selected in the slider. Further detail provided in the side graph shows comparisons between the trend for each status code against the actual count in the hour. The response time metric is displayed in the day view with a bullet chart of average response times in the hour. Additionally, a scatterplot shows minute by minute data across the hour for a more granular view of anomalous points.

Although the raw data contains 4 full weeks of log files, the system only displays the last 3 weeks of this data. This design choice allows the system to train over a full week. The training week is necessary because during the first week of data, the trend averages and standard deviations are not accurate enough to effectively classify anomalous datasets. As the system analyzes more and more data, its detection accuracy will increase.

7. RESULTS

User studies give insight into the success of our data visualization work. Multiple iterations of the product, with a user study after each version, provide a basis on which to compare different visualization techniques and styles. This allows for a comparison of benchmarks and survey responses between rounds, thereby turning subjective criteria into objective statistics. On each revision of the application we adjusted our approach to incorporate data from previous rounds to optimize our visualization.

The goal is to create a visual tool that is easily understood, so that even those with little technical background can efficiently interpret the data that is displayed. Thus, our user studies were conducted with a wide sample of people: those working in the telecommunications industry such as a Comcast employee, people with a technical background, and college students with no knowledge of the subject. Each user was isolated when interacting with the interface.

To give objective value to otherwise subjective user studies, we used the cognitive jogthrough method: a structured evaluation of a user’s interaction with the application. This method facilitated better quantitative and qualitative improvements to the UI in each iteration. Figure 16 shows an example worksheet that we use to evaluate comprehension and effectiveness of our application.

In the day view of the status codes metric, each hour shows a horizontal stacked graph that displays the count of each status code level from 200 to 500 in each hour selected in the slider. Further detail provided in the side graph shows comparisons between the trend for each status code against the actual count in the hour. The response time metric is displayed in the day view with a bullet chart of average response times in the hour. Additionally, a scatterplot shows minute by minute data across the hour for a more granular view of anomalous points.

Although the raw data contains 4 full weeks of log files, the system only displays the last 3 weeks of this data. This design choice allows the system to train over a full week. The training week is necessary because during the first week of data, the trend averages and standard deviations are not accurate enough to effectively classify anomalous datasets. As the system analyzes more and more data, its detection accuracy will increase.

7. RESULTS

User studies give insight into the success of our data visualization work. Multiple iterations of the product, with a user study after each version, provide a basis on which to compare different visualization techniques and styles. This allows for a comparison of benchmarks and survey responses between rounds, thereby turning subjective criteria into objective statistics. On each revision of the application we adjusted our approach to incorporate data from previous rounds to optimize our visualization.

The goal is to create a visual tool that is easily understood, so that even those with little technical background can efficiently interpret the data that is displayed. Thus, our user studies were conducted with a wide sample of people: those working in the telecommunications industry such as a Comcast employee, people with a technical background, and college students with no knowledge of the subject. Each user was isolated when interacting with the interface.

To give objective value to otherwise subjective user studies, we used the cognitive jogthrough method: a structured evaluation of a user’s interaction with the application. This method facilitated better quantitative and qualitative improvements to the UI in each iteration. Figure 16 shows an example worksheet that we use to evaluate comprehension and effectiveness of our application.

In the day view of the status codes metric, each hour shows a horizontal stacked graph that displays the count of each status code level from 200 to 500 in each hour selected in the slider. Further detail provided in the side graph shows comparisons between the trend for each status code against the actual count in the hour. The response time metric is displayed in the day view with a bullet chart of average response times in the hour. Additionally, a scatterplot shows minute by minute data across the hour for a more granular view of anomalous points.

Although the raw data contains 4 full weeks of log files, the system only displays the last 3 weeks of this data. This design choice allows the system to train over a full week. The training week is necessary because during the first week of data, the trend averages and standard deviations are not accurate enough to effectively classify anomalous datasets. As the system analyzes more and more data, its detection accuracy will increase.

7. RESULTS

User studies give insight into the success of our data visualization work. Multiple iterations of the product, with a user study after each version, provide a basis on which to compare different visualization techniques and styles. This allows for a comparison of benchmarks and survey responses between rounds, thereby turning subjective criteria into objective statistics. On each revision of the application we adjusted our approach to incorporate data from previous rounds to optimize our visualization.

The goal is to create a visual tool that is easily understood, so that even those with little technical background can efficiently interpret the data that is displayed. Thus, our user studies were conducted with a wide sample of people: those working in the telecommunications industry such as a Comcast employee, people with a technical background, and college students with no knowledge of the subject. Each user was isolated when interacting with the interface.

To give objective value to otherwise subjective user studies, we used the cognitive jogthrough method: a structured evaluation of a user’s interaction with the application. This method facilitated better quantitative and qualitative improvements to the UI in each iteration. Figure 16 shows an example worksheet that we use to evaluate comprehension and effectiveness of our application.

In the day view of the status codes metric, each hour shows a horizontal stacked graph that displays the count of each status code level from 200 to 500 in each hour selected in the slider. Further detail provided in the side graph shows comparisons between the trend for each status code against the actual count in the hour. The response time metric is displayed in the day view with a bullet chart of average response times in the hour. Additionally, a scatterplot shows minute by minute data across the hour for a more granular view of anomalous points.

Although the raw data contains 4 full weeks of log files, the system only displays the last 3 weeks of this data. This design choice allows the system to train over a full week. The training week is necessary because during the first week of data, the trend averages and standard deviations are not accurate enough to effectively classify anomalous datasets. As the system analyzes more and more data, its detection accuracy will increase.
what their next atomic action should be. This question is intended to illuminate the order in which the user interprets information and interacts with the UI. Once the user takes the action, the process is repeated to construct a flow chart of behavior.

Qualitative comments from users give us a sense of how users see the tool, for example, one user explains, “I see that there was a red bar at 4pm which definitely stands out and seems like it could be indicative of a problem.” From this we understand that our color scheme has been effective for at least one user. Another user explains the logic behind scrolling over all of the squares as their first step on the calendar page as a way to see, “if the pop ups give different information about each day. Also to see if there is a distinct difference between different colored squares.” This informs us that users were seeking more information on the exact significance of the colors. We then evaluate the tool based on these qualitative comments as well as with a quantitative analysis of how many users understand the task, how much they deviate from the right path, and the correctness of their interpretation. We define “right path” as the series of actions that we want the users to take, and “correctness” as the value extracted from the UI, such as identifying an anomaly or determining its cause.

Maintaining ignorance about the data to be seen is important to having controlled user studies. To mitigate this issue, there is a different outlier depicted by the user interface every iteration of the user study. Additionally, the infographic is focused on different subject areas as much as possible.

We conducted two full user studies, and one feedback session with a Comcast network operator. In the first user study, we used A/B testing along with the cognitive jogthrough method to compare two versions of our user interface. For this first user study, we focused on our goal of an easily understood tool with little training required by testing our application on nontechnical users. We used a specific day of data to avoid overwhelming the user with too much information to sift through. This had two benefits, it focused the user feedback on the actual visualizations and tools of interaction and allowed us to compare the ease of learning the tool between two versions of the UI without too many other factors affecting the metrics recorded. Group A evaluated a user interface focused on being clean and uncluttered to prevent overwhelming the user. Group B evaluated a user interface with a few helpful hints embedded on-screen, explaining how to use the tool. The B interface increased the busyness of the display slightly, but was found to have a positive impact overall. Those users interacting with B finished evaluating the UI 50% faster than those interacting with A. Additionally, 20% more of B discovered the anomaly in the data than A.

In our second user study, we used Computer Science college students and broadened the data to include all of the data for March that we had. This increase in data made the study more open ended with a less clear correct anomaly to be found, but gave a better understanding of our tool’s use overall. With the addition of the second user study we could compare the speed of understanding metric between studies. This can be seen in Figure 17. The increase in time to understand the tool between user study one’s group B and user study two is not surprising when considering the increased complexity having many days of data to analyze created for the user. The decrease in time to understand the tool between user study one group A and user study two is a promising result. It shows an increased understanding of the inherent tool after the improvements made to the UI.

![Figure 17: User Study Results](image)

Finally, in our feedback session with a Comcast network operator we received three main areas of feedback. He noted the ease of use for the tool without training or knowledge in the field. Additionally, he intentionally gave us no context for the network data we were given to see the conclusions we could find with our analysis. Through use of our tool, we discovered two interesting hypotheses about the network data, which he confirmed. The first hypothesis was that the server shutdown every day at 4pm UTC. Our Comcast contact confirmed a restart and redeploy is conducted every day at that time on that particular server. Secondly, we noticed that the weekend traffic was significantly higher than that of weekdays. We found this to be an unusual trend but the Comcast network operator explained the trend by telling us the data was focused on television usage which is higher on the weekends when people are home from work. He was impressed with the information that could be gleaned from the data using our visualizations. Finally, he agreed that this tool would be useful for Comcast network operators which validated our application and added to our conclusion that the system was successful.

8. ETHICS

The most significant ethical concerns when dealing with network data relate to the privacy of the users of the network. AnomVis has the capability to bring scrutiny to users’ behavior with the amount meaning it can extract from HTTP request data. This data holds a significant amount of private information including websites visited, search history, IP addresses and user identification. This information has the potential to provide companies like Comcast with great value, helping them to track trends and identify patterns in behavior amongst their users. It also helps them better understand their system usage and make adjustments accordingly. However all this information also raises the concern of “profiling, discrimination, exclusion, and loss of control.” [13]

There are privacy regulations regarding the use and storing of analyzed big data. The solution is de-identification of data: anonymizing user IDs to prevent breaching these
rules. However, it is also important to balance privacy with public safety. With huge companies like Comcast, their wide reach and the incredible quantities of network data they collect give them the ability to identify security threats and attacks on their servers which could affect millions of customers. AnomVis’s applications are limitless as it can be expanded and modified to visualize any network. It is key to keep in mind the ethical implications of its expanding use cases as an application that maintains big data trends.

9. Future Work

Looking forward, there are several areas of improvement and expansion for AnomVis. The first is decreasing the processing time of raw data. This can be accomplished both through the optimization of the processing algorithms and through the expansion of machine infrastructure. Once the processing time has been decreased to fractions of a minute, the application can be easily scaled up to monitor additional servers and datacenters.

The second area of expansion is integrating AnomVis into a large network in order to introduce real time processing into the application. This integration will allow human operators to immediately identify problems in the system as they are occurring and implement fixes to prevent further disruption. Once the data processing component of the system works in real time, only slight changes to the UI are required to dynamically update the visualizations at the new speed.

Another area of future work is adding more metrics to analyze. Overall request frequency per URL can be analyzed to determine if a particular URL is receiving an anomalous number of requests. Abnormalities in this area are indicative of possible Denial of Service attacks on the system. Additionally, requests based on Server IP Address can be analyzed to determine if a particular server is running slowly or is being overloaded.

Future implementations could incorporate more visualization research. Some additional cognitive metaphors this application could include involve motion, orientation, and annotations. Motion is often used to draw a user’s attention to an object. This movement can be used to highlight anomalies with bouncing, or expansion and contraction of, shapes on the screen. Orientation is the relative physical position or direction of something. Orientation can be further incorporated with visualizations by taking advantage of users’ inherent associations of up meaning better and down meaning worse. This visual could be accomplished by having normal data above a threshold and anomalies below.

Additionally, orientation can be used to de-emphasize aspects of the visualizations when they are no longer the main focus, either through rotation and motion or by increasing the distance of the visualization from the current focal point of the screen. Annotations add an extra dimension to user interaction by giving the user context with mouse movement. They appear in visualizations based on user actions that occur such as hover, scroll, or click, and take the form of tooltips, pop-ups, and help boxes. Tooltips show information when a user hovers and give more context without changing the current view while help boxes and pop-ups assist the user in using the tool. All of these cognitive metaphors would advance the goal of an easy to learn tool that suggests anomalies to the user without requiring them to parse through all data points looking for abnormalities.

10. Conclusions

AnomVis aims to help guide the direction of new industry standards towards user focused applications using anomaly detection techniques. It is a scalable and flexible application with many potential use cases including analyzing network packets and HTTP request data. With months and years of data, it will accurately reveal outliers and display real trends to network operators. These operators will then be able to identify problems and patterns in network usage more easily and ultimately locate any security threats and strengthen any weaknesses in the network.

11. References