UNIVERSITY OF VIRGINIA

MASTER OF SCIENCE THESIS

Characterizing University of Virginia's Baseline Network for Anomaly Detection

Author
Matthew RODRIGUEZ

Thesis Advisor
Dr. Donald BROWN

DEPARTMENT OF SYSTEMS AND INFORMATION ENGINEERING

Thesis submitted in fulfillment of the requirements for the degree of Master of Science

Contents

List of Figures

List of Tables

Abstract

The following paper details the research conducted for characterizing the University of Virginia's baseline network behavior. The study lays out a methodology for extracting network traffic, processing the captured data, reading the traffic into the utilized programs, and generating time series graphs for characterization analysis. Furthermore, the paper details the study's initial results from various models followed by data cleaning and layering techniques to achieve a final model in distinguishing the "pattern of life" of the network traffic. Differing from previous studies, this study aims to characterize a large network accessed by an academic population through analysis of Bro logs, specifically http.logs. Future work will focus on constructing wider time window comparisons of various features, in order to analyze how the baseline network behaves across each day, week, month, and conceivably year. The end result for this study is to develop a more accurate model in distinguishing the "pattern of life" of the University of Virginia's network traffic in order to detect network anomalies. This network characterization model would aid the University of Virginia and other large academic institutions in understanding typical behaviors of their network traffic and detecting anomalous traffic for intrusions.

1 Problem Statement

The cyber domain is the newest frontier in the Digital Age. The competition of owning superior cyber technology and dominating cyberspace attracts governments and businesses to obtain faster and cheaper forms of information and communication capabilities [1]; however, the increase in demand and reliance for this advanced technology comes with devastating vulnerabilities when exploited. This year, the United States listed cyber as their top threat to the stability of the country [2]. Adversaries are continuously entering the cyber domain to conduct espionage missions, attack critical economic systems, and exploit national information.

Within the era of growing cyber threats, the University of Virginia (UVA) has experienced its share of exploits from cyber attacks on its network. Between the years 2005 and 2007, hackers with Internet Protocol (IP) addresses traced from China breached UVA's network to gain access to a list of names and social security numbers of over 5,700 UVA faculty members [3]. More recently, in 2015 UVA was forced to shutdown its network in order to thwart another cyber attack originating in China [4].

As a response to the increasing cyber threats, UVA and much of academia has been seeking out advanced network intrusion systems to bolster their cybersecurity. Network characterization is important to UVA's faculty and students in providing an understanding about UVA's network usage as well as network user behaviors across particular time spans. In return, these details highlight typical traffic patterns to help distinguish a typical network behavior in order to compare against periods of anomalous traffic for intrusion detection.

2 Literature Review

In the past decades, cybersecurity has rapidly grown into a leading research field in the world. Researchers are constantly striving to out-pace their cyber adversaries in creating newer and more advanced cybersecurity measures. While there currently are multiple aspects associated with cybersecurity, this study focused on how to properly and accurately characterize UVA's baseline network behavior to aid in anomaly detection.

Characterization of network traffic involves deriving unique traffic classes and statistics in order to gain a better understanding of its behaviors. Characterization of network traffic is an expansive field consisting of numerous unique approaches depending on which behaviors of the network traffic researchers are attempting to characterize. This study centered around three main aspects of network characterization: workload and performances measurements, traffic prediction, and anomaly detection.

Previous studies in characterization of network traffic focus on understanding the workload and performances of their systems. Ersoz and et al. [5] characterized the general trends in the network behavior in their data center in terms of performance at each tier. Their study used real implementation of a clustered three-tier data center to analyze the arrival rate and inter-arrival time distribution of requests to individual server nodes, the network traffic between tiers, and the average size of messages exchanged between tiers. Although the study's objective was centered around performance measurements of their data center, the characterization techniques and analysis implemented in the study, create a proper baseline for network behavior using network request rate as a feature selected from Hypertext Transfer Protocol (http) connections. In addition, this is key in understanding network behaviors during specific high-demanding events especially for specific time intervals in this study.

While Ersoz and et al. monitored how traffic behavior differed across workloads, other studies took a passive approach at characterizing network traffic behavior. In an earlier study, Kotz and Essien [6] characterized the traffic behavior of Dartmouth College using network traces of their wireless traffic. The study monitored these traces to better understanding the load on the network infrastructure and the distribution of network usage across campus. Both Kotz and Essien's study and this study aim to characterize an academic institution's network traffic by better understanding typical network behavior throughout days and weeks. Both of studies highlighted unique approaches to characterizing their network traffic by monitoring workload and performances of their systems.

Characterization of network traffic for traffic prediction is an extensive research field. In the lanes of traffic prediction, an objective of this study is to use collected network traffic as a training set for a model to make predictions of future behaviors tested against the following week of network traffic. Joshi and Hadi [7] presented in their study various techniques to analyzing network traffic including neural network based techniques, data mining techniques, linear and non-linear models. The various techniques and models presented in their research are vital in characterization to better understand normal traffic patterns. Interestingly, their study highlighted how Yanhua Yu et al. research [8] demonstrated the use of autoregressive (AR) and autoregressive integrated moving average (ARIMA) models to model time series of network traffic for forecasting traffic. Their observed relative mean absolute percent error (MAPE) between prediction values and original values were about 2 percent. This study utilized similar approaches to AR and ARIMA models in order to achieve lower MAPE values for traffic predictions. In addition, their study demonstrated the use of generalized autoregressive conditional

heteroskedasticity (GARCH) models to capture the bursty nature of network traffic. GARCH models were used in this study to characterize the variance of the network traffic at each level of data manipulation. This study utilized various predictive modeling techniques at fitting weeks' worth of network traffic in order to forecast the next five days of traffic.

An early study, by Huang and et al. [9], characterized network behaviors using wavelet signal analysis for anomaly detection. Based on known models, current data can be evaluated against them to flag anomalies by leveraging pattern or signature matching features. However, in the case of an unknown anomaly, this approach is ineffective. The research focused on signal analysis of four classes of network traffic anomalies with the application of wavelet filters as an effective method of exposing ambient and anomalous traffic. This study's signal analysis leveraged time characteristics of IP flow and Single Network Management Protocol (SNMP) data collected at the border router over a six-month period which applied filtering techniques, in order to effectively expose local frequency details of anomalies. This initial analysis of network traffic revealed an different approach at network characterization by distinguishing typical network behavior using wavelet signals for anomaly detection. While this study doesn't test for anomaly detection using wavelet signal analysis, it is important to leverage time when characterizing network traffic. Based on the same idea of wavelet analysis for network prediction, Cardaso and Vieira [10] proposed an adaptive approach to estimate the energies of the wavelet and scale coefficients of the Haar wavelet transform used in the multifractal modeling of network traffic traces. Within their study, they used an autocorrelation function to update orthonormal basis functions in a fuzzy system. This method proved to provide lower error rates than previous wavelet analysis approaches for network prediction. This approach could be an appropriate

route if a linear trend can be distinguished from UVA's network traffic. In a more recent study, Price-Williams and Heard [11] characterized their network traffic by constructing a model of a computer's normal behaviour for anomaly detection. Their study presented a framework for characterizing the conditional intensity for normal, user-driven computer network behaviour. Their study highlighted the need to analyze time of day windows for possible patterns within network usage. While this study doesn't cover any anomaly detection techniques, the explained anomaly detection techniques displayed the importance and effectiveness of time in network characterization.

Due to the shear size of the time series data along with the periodic request arrival rate spikes, it is crucial to properly fit and test models that can capture the network traffic. Many recent studies approach these challenges with network traffic prediction with machine learning methods. Gamboa [12] provides a complete overview on how the application of Deep Learning techniques across recent studies have contributed and improved time series analysis. Previous classification methods relied heavily on the usage of domain specific features normally crafted manually by human. This made finding the best feature or combination of features laborious and quality of feature dependent. As an example to the advantages of machine learning techniques, Raford and et al. [13] proposed using Long Short-Term Memory (LSTM) cell Recurrent Neural Networks (RNN) to capture sequences of communications between computers on a network. This machine learning model was then used predict the communications between two IPs and to determine how typical or atyptical the observed communications are. Similar to how our dataset is not necessarily clean of intrusions since it is raw network traffic, their machine learning approach was able to achieve area under the ROC curve scores of 0.84 using proto-byte features and 0.74 using service port features. For the selected

features from our generated Bro logs, which models can provide the most accurate fitting and forecasting in order to distinguish typical behavior from the network traffic. Makridakis and et al. [14] proved in their study how traditional statistical models dominated across both accuracy measures and for all forecasting horizons examined. Their study compared the one-step forecasting results from eight classical methods (Naive 2, simple exponential smoothing, holt, damped exponential smoothing, average SES+holt+damped, theta method, ARIMA, and ETS) with eight machine learning methods (Multi-layer perceptron, Bayesian neural network, radial basis functions, kernel regression, K-nearest neighbnor regression, CART regression trees, support vector regression, and Gaussian processes) along with two modern neural network algorithms (Reccurent neural network and long short-term memory). Each model was provided the same Box-Cox transformed data series. From the one-step forecasting results, every classical model, except for Naive 2, outperformed the machine learning models. According to the MAPE measurement, the study's ETS model was able to forecast the testing set with an error value of 7.12 percent and the study's ARIMA model was able to forecast the testing set with an error value of 7.19 percent. The best performing machine learning model, Bayesian neural network, was able to forecast the testing set with an error value of 8.17 percent. While machine learning techniques are improving certain aspects of time series analysis and forecasting, the direction of this study focused on specific feature selection from Bro logs to create univariate time series. As a result, this study aims to utilize similar classical models due to their accuracy in one-step forecasting of univariate time series.

Following the studies conducted in characterizing network traffic, few approaches attempt to distinguish a typical network behavior or a "pattern of life" from network traffic. Derived from previous studies, "pattern of life," de-

scribes a recurrent way of acting by a population toward a given situation [15]. The study in this paper aims to utilize characterization techniques to depict and analyze UVA's baseline network behavior to distinguish the "pattern of life" of the network traffic for further studies in anomaly detection.

3 Research Objective

The objective for this research is to characterize a baseline network behavior of University of Virginia's network traffic by producing a model that can distinguish the "pattern of life". The "pattern of life" will provide a time of day and day of week characterization for better understanding typical network usage patterns by the UVA population. Once the "pattern of life" can be determined, further work will include establishing thresholds for anomaly detection on a large academic institution's network.

4 Characterization Methodology

This section discusses the methodology for characterizing UVA's baseline network behavior. The characterization methodology is divided into four phases:

1. Constructing a pipeline for data extraction from the network traffic.
2. Developing a script for reading extracted network logs.
3. Creating time series graphs to depict selected features of network traffic.
4. Designing a model based on the generated time series to distinguish the "pattern of life" of the network traffic.

Figure 1 provides an overview of the characterization methodology phases. Each phase will be discussed in detail about the progress this study has made in the characterization process.

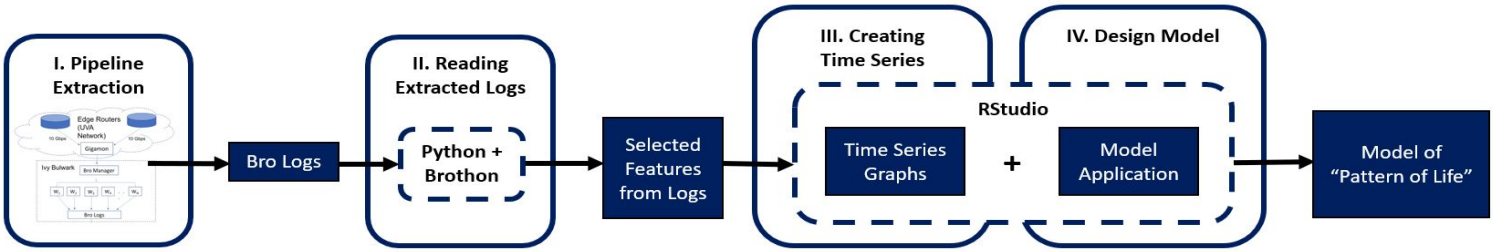

Figure 1: Overview of the characterization methodology phases.

4.1 Pipeline for Data Extraction

In order to process data from UVA's network traffic, the data must first be extracted. As shown in Figure 2, traffic at the speeds of up to 10 Gbps is transferred from the two-edge routers on UVA's network to the Gigamon network to be copied. Once traffic was copied within the Gigamon network, it was sent to be processed in Ivy Bulwark (UVA's secure computing environment). Within Ivy Bulwark, the extracted traffic was evenly distributed by the Bro Manager to each worker to be parsed based on network protocols. Bro Manager condensed these outputs into various bro logs based on protocols (conn.log, http.log, smtp.log, snmp.log, dns.log, ftp.log, etc.) for one-hour time windows. These Bro logs were further processed for anonymization to protect the sensitive information of users within UVA's network that was not needed for this study. For this study, only conn.logs and http.logs were condensed by the Bro Manager for feature selection and network characterization. The http.logs used in this study were from previous pipeline data extractions last year but exclude any fields that may contain sensitive information. The anonymized conn.logs (anon_conn.logs) and http.logs were then ready to be read by Python programming language for network characterization.

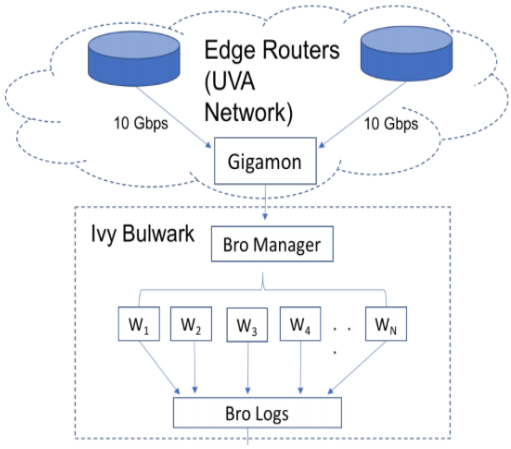

Figure 2: Visualization of Pipeline for Data Extraction

4.2 Reading Extracted Logs

Once the Bro logs were read and saved into a data frame, the research was able to explore specific features and characteristics of the captured network traffic; however, while trying to initially read in the anon_conn.logs, the research ran into three issues that users commonly experience when working with conn.logs in Python.

While there were packages for Python (i.e. Brothon [16]) that are able to read Bro logs, the packages were unable to read the anonymized conn.logs. In order to solve this first issue, a Python script was written that was able to successfully read in the anon_conn.logs. Although this was a simple fix, it allowed for the Bro logs to be read and to maintain their anonymity.

The second and third issues are regarding the shear size of each anon_conn.log. Compared to other Bro logs produced during the data extraction phase, a single hour conn.log (or anon_conn.log) can range from 0.75 to 1.0 GB in memory size with 22 different fields [17]. The second issue deals with the time it takes to successfully read in a conn.log and organize it into a manageable pandas

data. Currently, it takes about three hours to loop through and read an entire days worth of conn.logs for a single day's worth of traffic, which is significant. While this is not the fault of the Bro logs, it can be fixed by reading the conn.logs through a more advanced computer cluster, rather than a single, personal machine. Current potential solutions include utilizing a cloud-computing cluster through Amazon Web Services [18] or UVA's own Rivanna cluster [19].

As discussed in the previous paragraph, each one-hour anon_conn.log can range from 0.75 to 1.0 GB. This is where the third issue resides. If that amount of information is multiplied by 24 logs for each day and 18 to 24 GBs of data would have to be read in. In addition, once a log is successfully read and organized into a manageable pandas data frame, a single log can increase to about 60 GB. Many machines simply do not possess the Random Access Memory (RAM) required to mitigate the effects of the swelling size of the anon_conn.logs. In order to counter the size issue of each anon_conn.log, a Python script was written that read each one-hour anon_conn.log at a time, selects a desired feature for extraction, and bins each feature based on their timestamps down to the second. Although this technique limits the user to focusing on a single feature at a time, it enables the user to manage and work with the size of the study's anon_conn.logs.

4.3 Creating Time Series

Once the http.logs or anon_conn.logs are successfully read, the same Python script allows the user to select which features the user would like to retain before constructing the data frame. Features this study examined include the request and connection arrival rates, which were based on the characterization technique used in *Characterizing Network Traffic in a Cluster-based, Multi-tier Data Center*. In addition, the study focused on total content size and byte rates

features used in the characterization technique from *DISCLOSURE: Detecting Botnet Command and Control Servers Through Large-Scale NetFlow Analysis*. The http.logs used in this study focused on Request Arrival Rates and Total Content Size Rates from Server. For the anon_conn.logs, Connection Arrival Rates and Total Byte Rates from Response Source were focused on. As mentioned in the previous section, each feature extraction followed the same methodology in order to accommodate for the large size of a day's worth of logs.

To process the logs based on arrival rates, bins were created for each second based on the requests' or connections' timestamp. Once a bin was assigned, every request or connection was counted by their respective bins to give the total requests or connections for each corresponding bin. The sum of each bin for every second is now considered the arrival rate for requests or connections throughout that day.

For creating total content and total byte rates, a specific field was selected within the Python script that captured the data transferred. Within a processed http.log, the *response_body_len* field tracked the actual uncompressed content size of the data transferred from the server. For a processed conn.log (or anon_conn.log for this study), the *resp_bytes* field tracked the responder payload bytes. Once a data frame (including the timestamp field and their respective content/byte fields for each log) was constructed, a bin was assigned for each second based on the requests' or connections' timestamp. For every bin assigned, each requests' or connections' content/byte fields were summed up by their corresponding bin. Once a data frame was created for the four features, each was then saved into a .csv file to be read into R-Studio in order to construct the time series graphs needed for network characterization.

This study included time series graphs that were created using R-Studio following Shumway and Stoffer's *Time Series Analysis and Its Applications with*

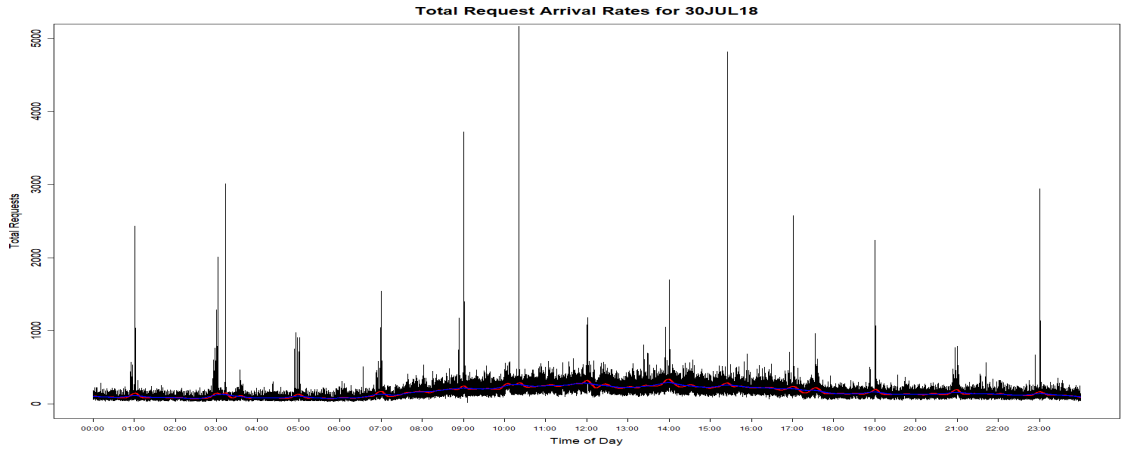

Figure 3: Time Series Graph of Request Arrival Rates for the full day July 30, 2018.

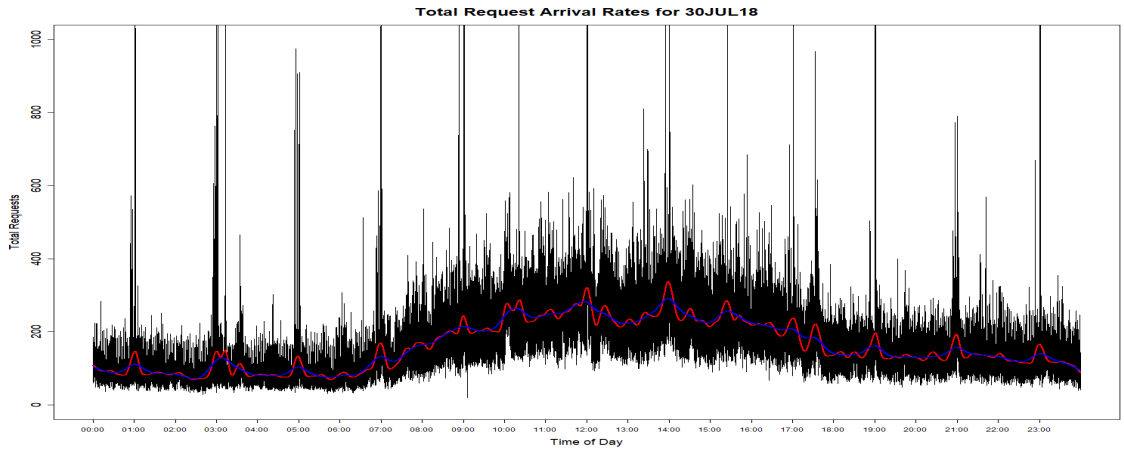

Figure 4: Time Series Graph of Request Arrival Rates for the full day of July 30, 2018 with limits set.

R Examples teachings [20]. The scripts for R-Studio that were written also followed the same general methodology. The .csv files containing the various data frames that were created in Python during the reading extracted logs and feature extraction phases were read into R-Studio. Next, the specified data frame was converted into a time series data frame including the selected feature, a

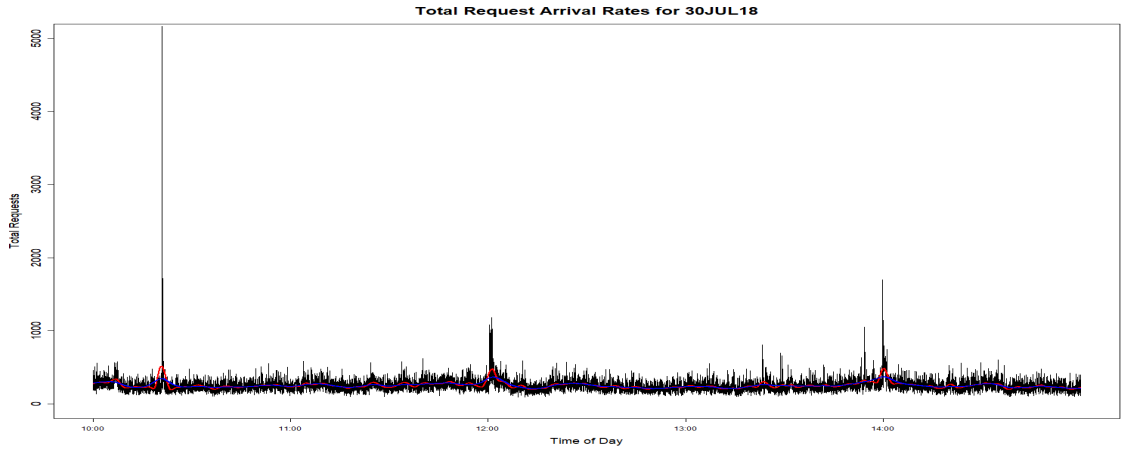

Figure 5: Time Series Graph of Request Arrival Rates for peak hours of July 30, 2018.

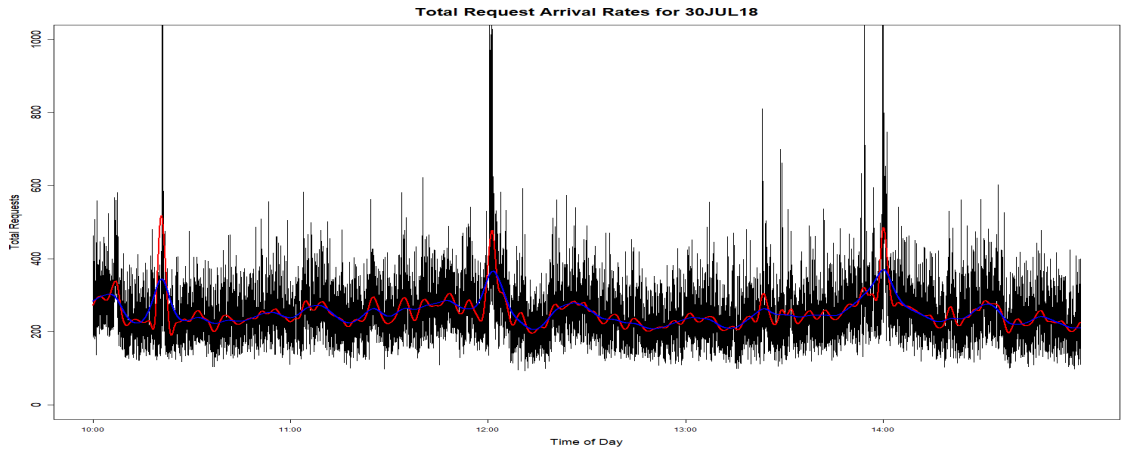

Figure 6: Time Series Graph of Request Arrival Rates for peak hours of July 30, 2018 with limits set.

start, and a frequency. From here, a time series plot was generated to depict the selected features of network traffic for network characterization.

Examples of generated time series are provided to display various time windows of network traffic. Figure 3, Figure 4, Figure 5, and Figure 6 are the generated time series covering the request arrival rates of network traffic on

July 30, 2018. Figure 3 and Figure 4 display time window of the entire day. Whereas, Figure 5 and Figure 6 display a five hour time window of the day's peak request arrival rates. All four figures include smoothing lines to help highlight patterns of increasing and decreasing request arrival rates throughout the network traffic. Furthermore, Figure 12 in the Appendix displays a month's worth (July 29 through August 25, 2018) of request arrival rates of network traffic to help compare each day of the week and each week. Based on these generated time series, we can begin to visually compare time of day and day of week behaviors. Varying time windows visually affect how certain behaviors are perceived. This is evident when comparing Figure 3 and Figure 4 for time of day analysis. When examining Figure 5 and Figure 6, we notice how these spikes of exceedingly large request arrival rates influence the smoothing lines. These periodic spikes will be further discussed and solved in Section 5 of the study.

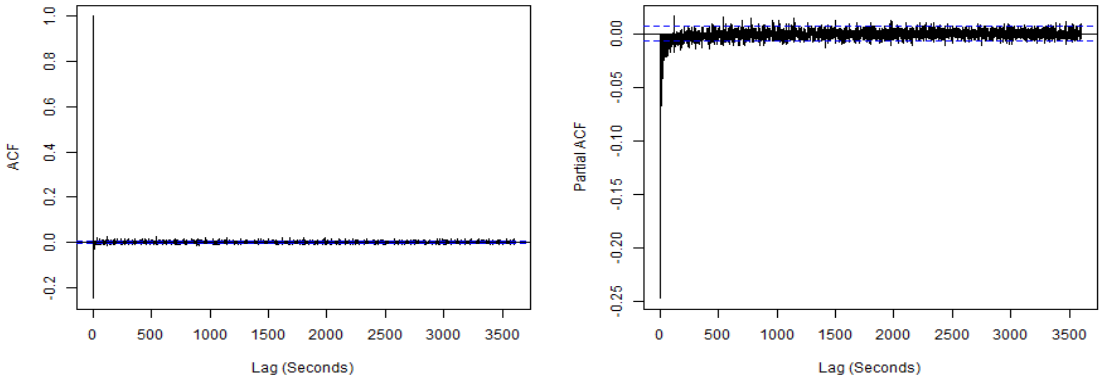

Figure 7: Auto-correlation and Partial Auto-correlation plots for July 30, 2018 time series .

Figure 7 is the Auto-correlation and Partial Auto-correlation functions with a lag extending to 3600 for July 30, 2018 time series data after differencing to

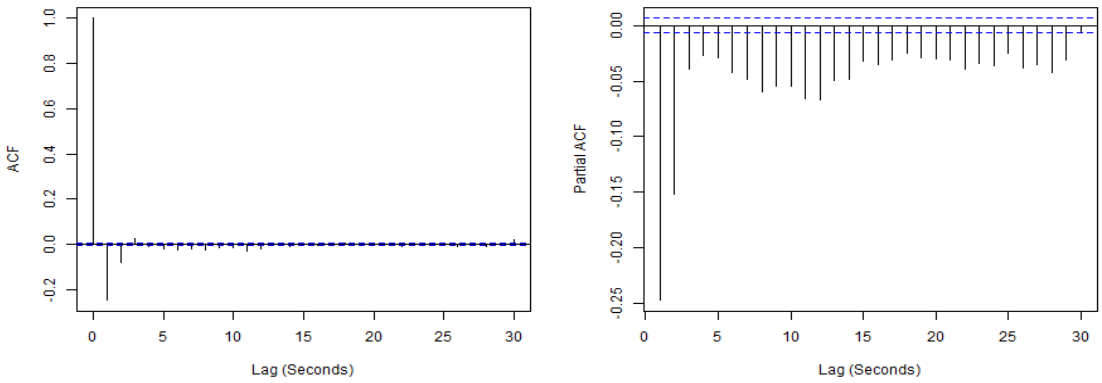

Figure 8: Zoomed in Auto-correlation and Partial Auto-correlation plots for July 30, 2018 time series .

remove any trend within the data. Figure 8 is the Auto-correlation and Partial Auto-correlation functions with a lag extending to 30 for July 30, 2018 time series data after differencing to remove any trend within the data. The ACF and PACF in Figure 8 both show a negative correlation at lag 1s, whereas the PACF in Figure 7 is tailing off. In the PACF in Figure 7, we noticed a significant positive correlation at lag 120s. This lag is not the same as the periodic spikes observed in the time series that occur every two hours. The lag at 120s could be a correlation related to high-frequency noise every 120 seconds. The ACF and PACF imply a simple MA(1) can be used for characterizing this time series. The next phase defines our models' design and objectives in characterizing a baseline behavior from these time of day and day of week comparisons in order to distinguish the "pattern of life" of UVA's network traffic.

4.4 Baseline Modeling

The purpose of the final model is to distinguish the "pattern of life" of the network traffic according to the selected features represented in the generated

time series. The model aims to identify a baseline network behavior by fitting a training and testing set from a month's worth of network traffic. Based on the trained model, researchers will be able to identify significant patterns to distinguish the "pattern of life" of the network traffic.

This section details the initial attempts at fitting models to the training set and provides the results of their fit and replication of the test set. This section then explains a solution to the request arrival rates spikes and the attempts at fitting the same initial models for a comparison in performance. Lastly, this section provides a final model that properly identifies significant patterns to distinguish the "pattern of life" of the network traffic.

4.4.1 Initial Models

For this study, three different smoothing models were used to filter and forecast the generated time series data in order to produce the initial results in distinguishing the "pattern of life." Each model was trained on the first three weeks (July 29 through August 18, 2018) of the https.logs request arrival rates data. The training set accounted for about 80.8 percent of the overall data. The trained models were then tested against the remaining five days of data (August 19 through August 23, 2018) of the https.logs request arrival rates data. The last two days of the week were dropped during the model design phase due to missing data as seen in Figure 15 and on. The test set accounted for about 19.2 percent of the overall data. The validation of each model was tracked by how accurately it forecast network behavior based on the test set.

The study's first technique was a simple exponential smoothing [21], which is ideal when working with data with no trend or seasonal pattern. Simple exponential smoothing weighs the recent data points more heavily than older data points in order to calculate forecasted values. The weight utilized was

determined by the smoothing parameter α, where $0 < \alpha \leq 1$, to help calculate the predicted values, $\hat{y}_{t+1}$ using the equation:

$$\hat{y}_{t+1} = \alpha y_t + \alpha(1-\alpha)y_{t+1} + ... + \alpha(1-\alpha)^{t-1}y_1$$

The greater the value of α, the more weight the equation gives to the most recent observation, resulting in recent changes in the data having a greater impact on the forecast values. The value of α was optimized by minimizing the test root mean square error (RMSE) value. While this model was complex enough for the size of data the study worked with, it provided an initial result in understanding significant network patterns.

The additive model [22] calculates forecast values using three variables, level (L_t), trend (T_t), and season (S_t), following the equation:

$$\hat{y}_{t+1} = L_t + kT_t + S_{t+k-m}$$

Each variable is updated through further equations utilizing three smoothing parameters, α for level, β for trend, and γ for seasonality, each constrained between 0 and 1.

$$L_t = \alpha(y_t - S_{t-m}) + (1-\alpha)(L_{t-1} + T_{t-1}),$$

$$T_t = \beta(L_t - L_{t-1}) + (1-\beta)T_{t-1},$$

$$S_t = \gamma(y_t - L_t) + (1-\gamma)S_{t-m}.$$

Since the study did not track any trend or seasonality with the data, the study specified the model as "ANN," where error was additive (A), trend was none (N), and seasonality was none (N). Overall, the model was more complex than

the simple exponential smoothing and should provide better accuracy with forecasting.

The multiplicative model [23] calculated forecast values using three variables that included level (L_t), trend (T_t), and season (S_t), by using the following the equation:

$$\hat{y}_{t+1} = (L_t + kT_t)S_{t+k-m}$$

Each variable was updated through further equations utilizing the same three smoothing parameters that included α for level, β for trend, and γ for seasonality, each constrained between 0 and 1.

$$L_t = \alpha y_t / S_{t-m} + (1-\alpha)(L_{t-1} + T_{t-1}),$$

$$T_t = \beta(L_t - L_{t-1}) + (1-\beta)T_{t-1},$$

$$S_t = \gamma(y_t / L_t) + (1-\gamma)S_{t-m}.$$

Once again, the study did not track any trend or seasonality with the data, the study specified the model as "MNN," where error was multiplicative (M), trend was none (N), and seasonality was none (N). Even checking with optimal model settings at "ZZZ," the multiplicative model at "MNN," was the optimal model selection.

Table 1: Selected Models and Parameters

Selected Models	**Model Parameters**		
	Smoothing, α	***Initial State,*** l	σ
Simple Exponential	0.09	-3.69	57.9548
Additive (A,N,N)	0.6903	111.5486	53.8636
Multiplicative (M,N,N)	0.4712	114.5432	0.3233

4.4.2 Initial Results

Table 2: Simple Exponential Smoothing Accuracy Measurements

Measurements	**Simple Exponential**	
	Training Set	***Test Set***
ME	2.27e-05	-2.12e-02
RMSE	57.95	46.99
MAE	34.29	32.86
MPE	NaN	-Inf
MAPE	Inf	Inf
MASE	0.64	0.61
ACF1	-0.22	-0.34

Table 3: Additive Model Accuracy Measurements

Measurements	**Additive Model (A,N,N)**	
	Training Set	***Test Set***
ME	-2.98e-05	7.91e+01
RMSE	53.86	111.17
MAE	30.57	81.05
MPE	-4.59	35.92
MAPE	20.98	46.28
MASE	0.93	2.47
ACF1	0.05	0.82

Table 4: Multiplicative Model Accuracy Measurements

Measurements	**Multiplicative Model (M,N,N)**	
	Training Set	***Test Set***
ME	-4.91e-05	8.07e+01
RMSE	55.13	112.34
MAE	29.94	82.41
MPE	-5.07	37.34
MAPE	20.49	47.09
MASE	0.91	2.51
ACF1	0.25	0.82

Proper predictive accuracy is most concerned with minimizing the RMSE and mean absolute percentage error (MAPE) values. While all three of the study's models yield low accuracy values in both RMSE and MAPE measurements, the study's additive and multiplicative models were able to properly fit and predict network traffic behavior. According to the MAPE measurement in Table 3, the study's additive model was able to fit the training set network data with an error value of 20.98 percent and predict the testing set network data with an error value of 46.28 percent. Whereas in Table 4, the multiplicative model was able to fit the training set network data with an error value of 20.49 percent and predict the testing set network data with an error value of 47.09 percent. A visual representation of the each model's predicting performance can be seen in Figure 15, Figure 16, and Figure 17 located in the Appendix.

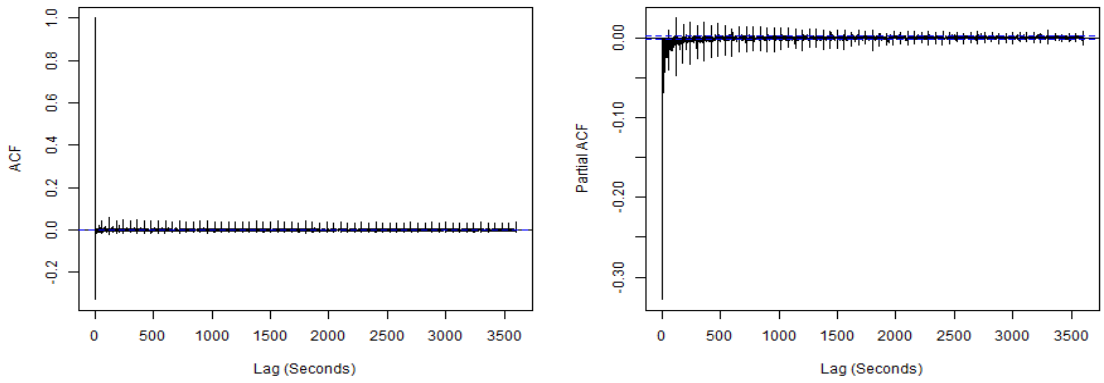

Figure 9: Auto-correlation and Partial Auto-correlation plots of July 29, 2018 to August 14, 2018 time series.

As anticipated in the figures prior, the periodic spikes are creating challenges for our models to properly fit and predict the time series data. A seasonal ARIMA (SARIMA) model was attempted at accounting for the periodic spikes. Initially, the training set consisted of the first twelve days of the time se-

ries data with the test set comprised of the following five days. This time window was chosen due to the fact the periodic spikes cease part way on August 15, 2018. The time series used for the ACF and PACF plots was log-transformed to stabilize the variance then differenced to remove any trend. In the ACF in Figure 9, there is a significant correlation at lag 1s, whereas the PACF in Figure 9 is tailing off. However, we noticed how there are periodic significant lags in both ACF and PACF plots in Figure 9. In ACF in Figure 9, we noticed a significant positive correlation at lags 120s. These lags are not the same as the spikes we observed in the time series at every two hours. The lags at every 120s could be a correlation related to high-frequency noise occurs every 120s, 240s, 360s, 480s, etc. The PACF in Figure 9 displays the same correlation every 120s, 240s, 360s, 480s, etc., that could be related to high-frequency noise. The ACF and PACF imply a simple MA(1) can be used for characterizing this time series. Unfortunately, when it came time to running the SARIMA(1,1,1, 0,0,1, lag = 86400), the function has a cap lag of 350 or runs out of memory. Hyndman [24] proposed using a Fourier series approach where the seasonal pattern is modelled using Fourier terms with short-term time series dynamics allowed in the error. Parameters of the best fit ARIMA model were determined by an auto-ARIMA function used in RStudio. Reinforcing the MA(1) that was suggested in the ACF and PACF in Figure 9, the best fit model was an ARIMA(4,2,1). The value of parameter K for the Fourier series approach was selected by minimizing AIC of the fitted model. Due to the memory size error, a single day of time series data was used to fit the model using a frequency of 3600 or periods of one hour. As noted by Hyndman, "The only real disadvantage (compared to a seasonal ARIMA model) that I can think of is that the seasonality is assumed to be fixed — the pattern is not allowed to change over time. But in practice, seasonality is usually remarkably constant so this is not a big disadvantage except

for very long time series." According to the MAPE measurement, the study's ARIMA model with Fourier series was able to fit the training set of network traffic with an error value of 21.66 percent. Unfortunately, this method did not prove more effective than the additive and multiplicative models at fitting to the data with periodic spikes. The size of the data and the length of the seasonality of the periodic spikes made it difficult to properly fit a model to the dataset. A solution to the periodic spikes is provided later in the study.

At each step of baseline modeling, each adjusted time series data is modeled to examine the changing variance. A lacking characteristic of our models on properly modeling univariate time series data is modeling the change in the variance over time. The study implemented the generalized autoregressive conditional heteroskedasticity (GARCH) process to model the change in variance correlated over time. A GARCH(p, q) model uses values of the past squared observations and past variances to model the variance at time t, following the equation:

$$\sigma_t^2 = \alpha_0 + \alpha_1 y_{t-1}^2 + \beta_1 \sigma_{t-1}^2$$

In order to determine the settings for the p and q parameters of the GARCH model, an auto-ARIMA function was used in RStudio. A GARCH(5,1) model was fitted to the time series data to examine the change in variance across time for the time series with data cleaning. Table 5 displays the coefficients estimated of the GARCH(5,1) model. The model produced p-values suggesting the α_0 and α_1 coefficients for this GARCH(5,1) are a strong fit for the data. Focusing on the diagnostic tests located in Table 6, the Jarque Bera Test and Box-Ljung Test confirm the strength of the coefficients by testing the residuals of the time series data. According to the Jarque Bera Test, the large chi-squared value and the p-value is less than 0.05 indicates the residuals are normally dis-

tributed. Since the p-value is less than 0.05 in the Box-Ljung Test, we can reject the hypothesis that the autocorrelation of residuals is different from 0. This low p-value suggests the model performs poorly at fitting to the data.

Table 5: GARCH(5,1) Model Coefficients

Coefficients	**Estimate**	**Standard Error**	**t value**	**Pr(>| t |)**
α_0	7.276e+03	2.352e+01	309.386	$<$ 2e-16 ***
α_1	5.692e-01	1.304e-03	436.338	$<$ 2e-16 ***
β_1	3.760e-11	2.542e-03	0.000	1
β_2	3.030e-02	2.556e-03	11.857	$<$ 2e-16 ***
β_3	2.261e-02	2.437e-03	9.276	$<$ 2e-16 ***
β_4	2.417e-02	2.385e-03	10.133	$<$ 2e-16 ***
β_5	2.740e-02	1.730e-03	15.836	$<$ 2e-16 ***

Table 6: GARCH(5,1) Model Diagnostic Tests

Diagnostic Test	**Data**	χ^2	**df**	**p-value**
Jarque Bera Test	Residuals	2.125e+10	2	$<$ 2.2e-16
Box-Ljung Test	Squared Residuals	822.24	1	$<$ 2.2e-16

The study's initial results show that it is possible for models to be fitted to a large amount of data and are still capable of distinguishing a significant network traffic behavior. This shows that a "pattern of life" can be distinguished given enough data to properly train the model; however, accurate network traffic behavior fitting and prediction of the test set was limited by the study's initial models' complexity and periodic request spikes. In order to overcome both of these challenges of properly training models to the network traffic, the study continues with cleaning of the data and fitting more complex models.

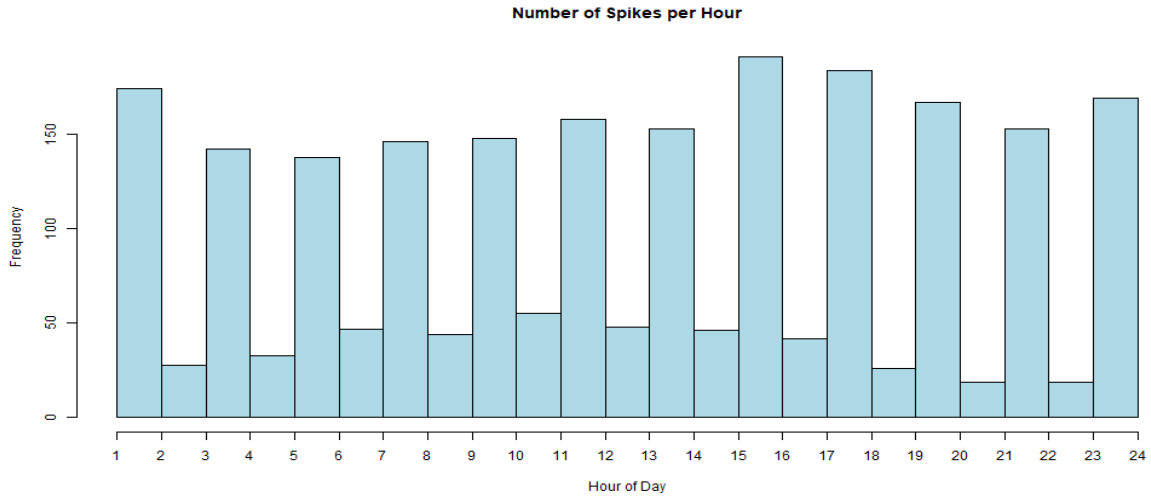

Figure 10: Histogram of spike frequency across each hour of the day

4.4.3 Cleaning of Data

The study previously noted the exceedingly large request arrival rates spikes when examining Figure 3 and Figure 5 and their affects on fitting observed in Figure 4 and Figure 6. Upon further investigation, these spikes appear periodic throughout each day. Figure 10 displays a histogram of spike frequency across each hour of the day. These spikes occur more frequently every other hour.

While Figure 10 provides a general understanding of the characteristics of the periodic spikes, a Fast Fourier Transform (FFT) helped characterize the data further. The FFT converts the time series data to the frequency domain to better characterize patterns. Figure 11 displays a FFT of the time series data from July 29, 2018 to August 8, 2018. From Figure 11, there exists two dominating frequencies: a low-frequency offset on the far left and a high-frequency noise on the far right. The low-frequency offset describes an existing behavior in the time series data over a long duration of time. The high-frequency noise describes an existing behavior of white noise along the overall behavior of the

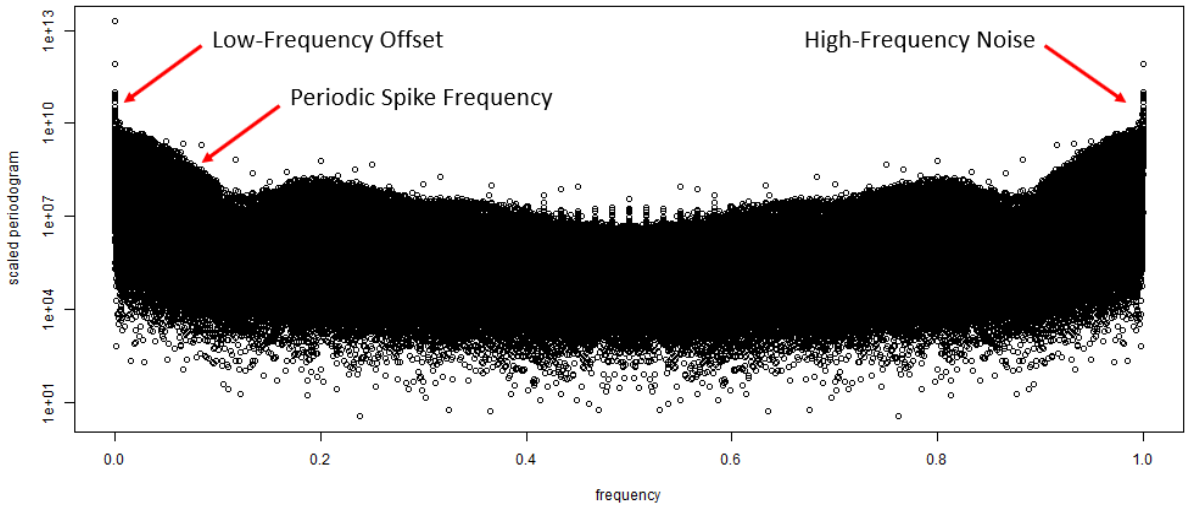

Figure 11: Fast Fourier Transform plot of the time series data from July 29, 2018 to August 8, 2018.

time series data. The next step was to characterize the periodic spikes by locating their frequency along the FFT. Our initial assumption was that the spikes followed a relatively low frequency due to the time between each spike. Each spike was subsetted then the difference between their time was calculated. The average of that calculation was taken to determine their average rate. One was divided by the average rate to find the frequency of the spikes: 0.449. After looking at the dataset of the spikes, it became apparent that each spike shown in the time series is actually a several second cluster of spikes. This caused the frequency to be different than our initial assumption. However, the study further explored the time intervals between each cluster. The frequency of these spike clusters that we see as just single spikes in the time series is 0.113. This frequency aligns with our initial assumption and helps characterize the periodic spikes as a relatively low frequency time series data.

In the Appendix, Figure 18 provides a time series graph of any request arrival rate greater than 1000 total requests per second to help observe the over-

all frequency of the spikes across the dataset. The spikes abruptly end before Thursday, August 16, 2018. Further investigation by UVA's network engineer yielded zero insight as to what could be the cause or end of these periodic spikes. Due to the affects of the spikes on training our models and the nature of the abrupt ending partway through the dataset, the study found a solution to handle these spikes.

In order to reduce the affects of the spikes on training our models, this study utilized the following data cleaning technique.

$$y_t - y_t - 1 >= \phi, \text{ then set } y_t = y_t - 1$$

Following this equation, the data cleaning technique took varying thresholds to reduce the affects of the spikes. For this study, the threshold was set to 100 total requests per second. The effectiveness of the data cleaning technique can be observed in Figure 19 located in the Appendix. Following the data cleaning, the study validated the effectiveness of the data cleaning technique on the spikes by retraining the initial models.

Using the data cleaning technique, an improvement in model performance is seen across all three models. Table 7, Table 8, and Table 9 displays the performance measurements for all three models trained on the cleaned data. According to the MAPE measurement, the study's additive model was able to fit the training set of network traffic with an error value of 18.88 percent and predict the testing set of network traffic with an error value of 46.81 percent. While there was no improvement in the test set, there is about a 2 percent decrease in error when fitting the additive model to the training set of network traffic. The multiplicative model was able to fit the training set of network traffic with an error value of 18.91 percent and predict the testing set of network traffic with an error value of 46.79 percent. Unlike the additive model, there

is slight improvement in both training and testing sets for the multiplicative model. Although it was not as drastic of an improvement was predicted, the data cleaning technique still helped better fit our models for distinguish the "pattern of life" of the network traffic.

Table 7: Simple Exponential Smoothing Accuracy Measurements on Cleaned Data

Measurements	**Simple Exponential**	
	Training Set	***Test Set***
ME	2.27e-05	-2.12e-02
RMSE	37.87	37.50
MAE	28.01	27.83
MPE	NaN	-Inf
MAPE	Inf	Inf
MASE	0.64	0.63
ACF1	-0.31	-0.31

Table 8: Additive Model Accuracy Measurements on Cleaned Data

Measurements	**Additive Model (A,N,N)**	
	Training Set	***Test Set***
ME	-5.53e-05	7.61e+01
RMSE	33.12	103.09
MAE	24.69	77.71
MPE	-4.39	36.98
MAPE	18.88	46.81
MASE	0.92	2.90
ACF1	0.13	0.85

Again, an auto-ARIMA function helped determine the p and q parameters for our GARCH model. A GARCH(2,1) model was fitted to the cleaned time series data to examine the change in variance across time for the time series with cleaning. Table 10 displays the coefficients estimated of the GARCH(2,1) model. The model produced p-values suggesting the α_0, α_1, and β_2 coefficients for this GARCH(2,1) were strong fit for the data. Focusing on the diagnostic

Table 9: Multiplicative Model Accuracy Measurements on Cleaned Data

Measurements	**Multiplicative Model (M,N,N)**	
	Training Set	***Test Set***
ME	-6.62e-05	7.64e+01
RMSE	33.17	103.34
MAE	24.72	77.99
MPE	-4.55	37.27
MAPE	18.91	46.79
MASE	0.93	2.92
ACF1	0.18	0.85

tests located in Table 11, the Jarque Bera Test and Box-Ljung Test confirm the strength of the coefficients by testing the residuals of the time series data. According to the Jarque Bera Test, the large chi-squared value and the p-value is less than 0.05 indicates the residuals are normally distributed. Since the p-value is less than 0.05 in the Box-Ljung Test, we can reject the hypothesis that the autocorrelation of residuals is different from 0. This low p-value suggests the model performs effectively at fitting to the data.

Table 10: GARCH(2,1) Model Coefficients

Coefficients	**Estimate**	**Standard Error**	**t value**	**Pr(>| t |)**
α_0	4.183e+03	6.932e+01	60.34	$<$ 2e-16 ***
α_1	7.142e-01	7.101e-03	100.57	$<$ 2e-16 ***
β_1	4.682e-11	8.305e-03	0.00	1
β_2	8.689e-02	7.658e-03	11.35	$<$ 2e-16 ***

Table 11: GARCH(2,1) Model Diagnostic Tests

Diagnostic Test	**Data**	χ^2	**df**	**p-value**
Jarque Bera Test	Residuals	84225	2	$<$ 2.2e-16
Box-Ljung Test	Squared Residuals	18.801	1	$<$ 1.451e-05

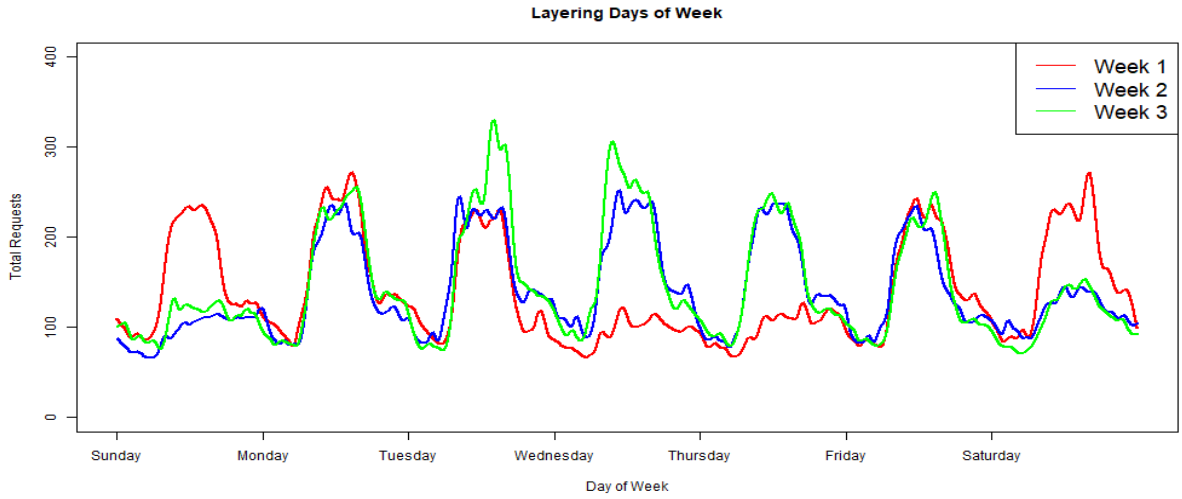

Figure 12: Layering of each day for each week with smooth lines.

4.4.4 Baseline Models

One of the main objectives of the baseline model was to identify significant patterns to distinguish the "pattern of life" of the network traffic. In order to meet this goal, the study aimed to leverage significant patterns in the time of day and the day of week. Figure 12 provides a time series display of layering each week of the training set aided by smoothing lines. By layering the three weeks of cleaned data used in the training set, patterns in the time of day and the day of week can be fitted more effectively than individually fitting each week by the models. The average of all the layers was calculated to help identify typical behaviors in the time of day and the day of week for the training set. Each of our three initial models were then fitted to this new layered training set to determine a baseline for typical network behavior across the time of day and the day of week. As seen in Table 12, Table 13, and Table 14, this layering method increased both model fitting and testing performance in determining baseline network behavior. According to the MAPE measurement, the study's addi-

tive model fitted the layered network traffic training set with an error value of 10.95 percent and predicted the testing set of network traffic with an error value of 42.41 percent. The study's multiplicative model fitted the layered network traffic training set with an error value of 10.95 percent and predicted the testing set of network traffic with an error value of 42.26 percent. The models trained on the layered network traffic fitted and predicted more effectively than the previous initial models in identifying baseline network patterns.

Table 12: Simple Exponential Smoothing Accuracy Measurements on Layered Network Traffic

Measurements	**Simple Exponential**	
	Training Set	***Test Set***
ME	-3.27e-05	1.98e-01
RMSE	21.19	37.50
MAE	16.28	27.83
MPE	NaN	Inf
MAPE	Inf	Inf
MASE	0.62	1.06
ACF1	-0.30	-0.31

Table 13: Additive Model Accuracy Measurements on Layered Network Traffic

Measurements	**Additive Model (A,N,N)**	
	Training Set	***Test Set***
ME	-5.15e-05	6.66e+01
RMSE	19.28	96.33
MAE	14.80	70.30
MPE	-1.50	28.72
MAPE	10.95	42.41
MASE	0.91	4.34
ACF1	0.13	0.85

Again, an auto-ARIMA function helped determine the p and q parameters for our GARCH model. A GARCH(2,1) model is fitted to the layered time series data to examine the change in variance across time for the newly layered

Table 14: Multiplicative Model Accuracy Measurements on Layered Network Traffic

Measurements	**Multiplicative Model (M,N,N)**	
	Training Set	***Test Set***
ME	-5.48e-05	6.62e+01
RMSE	19.29	96.05
MAE	14.80	70.01
MPE	-1.53	28.37
MAPE	10.95	42.26
MASE	0.91	4.32
ACF1	0.16	0.85

network traffic. Table 15 displays the coefficients estimated of the GARCH(2,1) model. The model produced p-values suggesting all four coefficients for this GARCH(2,1) are a strong fit for the data. Focusing on the diagnostic tests located in Table 16, the Jarque Bera Test and Box-Ljung Test confirm the strength of the coefficients by testing the residuals of the time series data. According to the Jarque Bera Test, the large chi-squared value and the p-value is less than 0.05 indicates the residuals are normally distributed. Since the p-value is less than 0.05 in the Box-Ljung Test, we can reject the hypothesis that the autocorrelation of residuals is different from 0. This low p-value suggests the model performs poorly at fitting to the data.

Table 15: GARCH(2,1) Model Coefficients

Coefficients	**Estimate**	**Standard Error**	**t value**	**Pr(**$>\mid t \mid$**)**
α_0	0.208125	0.102054	2.039	$<$ 0.0414 *
α_1	0.413584	0.008701	47.533	$<$ 2e-16 ***
β_1	0.198432	0.024723	8.026	$<$ 1.11e-15 ***
β_1	0.395589	0.018734	21.116	$<$ 2e-16 ***

As discussed previously, accurate network traffic behavior fitting and prediction of the test set was limited by the study's initial models' complexity

Table 16: GARCH(2,1) Model Diagnostic Tests

Diagnostic Test	**Data**	χ^2	**df**	**p-value**
Jarque Bera Test	Residuals	6495000	2	$<$ 2.2e-16
Box-Ljung Test	Squared Residuals	5180.6	1	$<$ 2.2e-16

and periodic request arrival rates spikes. The study provided a solution to the periodic request arrival rate spikes and a method for layering the training set to better leverage the time of day and the day of week. While, the initial models provided the study with a benchmark for the improvements made in distinguishing the "pattern of life" of UVA's network traffic, the issue of model complexity needed to be addressed. The study continued by applying an Auto-Regressive Integrated Moving Average (ARIMA) model to test the effectiveness of increased model complexity on distinguishing the "pattern of life" of UVA's network traffic.

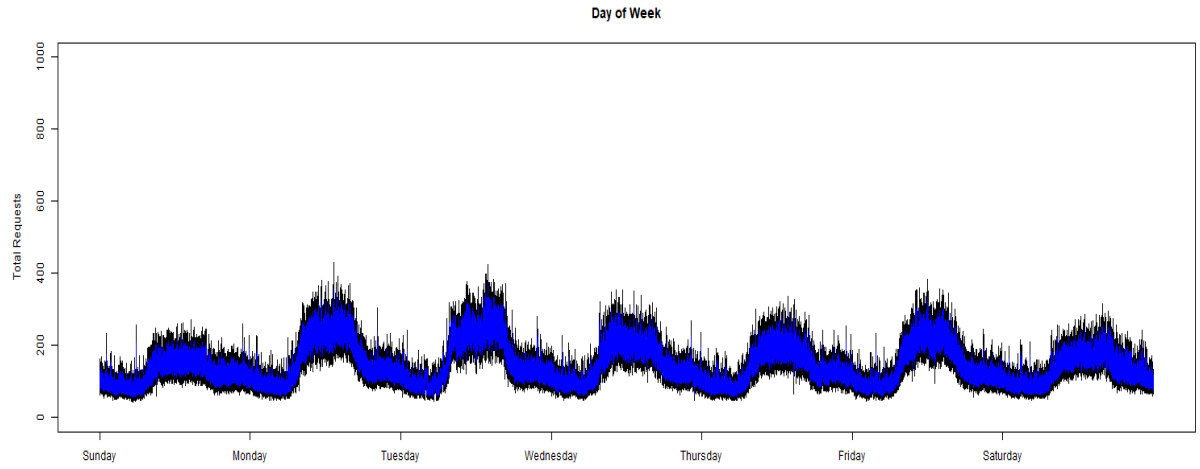

Figure 13: ARIMA Model fit to day of week.

The study implemented an ARIMA approximation within R-Studio to de-

Table 17: ARIMA Model Accuracy Measurements on Layered Network Traffic

Measurements	**ARIMA Model (2,1,2)**	
	Training Set	***Test Set***
ME	-6.33e-05	5.51e+01
RMSE	18.39	88.72
MAE	14.15	63.06
MPE	-1.66	18.60
MAPE	10.51	39.48
MASE	0.87	3.89
ACF1	0.00	0.85

termine the best fitting model to the layered weekly traffic training set. The ARIMA approximation results found an ARIMA (2, 1, 2) model with the coefficients *ar1* = 1.2706, *ar2* = -0.3295, *ma1* = -1.7867, and *ma2* = 0.7888 to construct the best fit to the layered weekly traffic training set. Figure 13 provides a visual display of the ARIMA model's fitting accuracy in distinguishing the "pattern of life." Located in the Appendix, Figure 20 displays the ARIMA model's fitting accuracy of the layered weekly network traffic across each individual week of network traffic. Visually, the ARIMA model's layered fitting accuracy appears to follow the typical behavior across each day of the week and each individual week of the network traffic. In addition to it's visual effectiveness, the ARIMA model performs better than the initial models when comparing actual measurements. According to the MAPE measurement in Table 17, the ARIMA model fitted the layered network traffic training set with an error value of 10.51 percent and predicted the network traffic testing set with an error value of 39.48 percent. Compared to the results of each week when not layered, the ARIMA model predicted Week 1 with an error value of 35.20 percent, Week 2 with an error value of 34.61 percent, and Week 3 with an error value of 35.23 percent. The ARIMA model fitted to the layered network traffic training set distinguished the "pattern of life" with a MAPE of only 39.48 percent. This is a

significant increase in determining typical network behavior compared to the attempts made by the studies initial models. Within the accepted error rate of the distinguished "pattern of life," the study is able to determine a baseline network behavior for any given hour of the day and day of the week. Any traffic outside of this accepted error rate can be deemed as not typical to the baseline network behavior of UVA's network usage. Future work can help decrease the error rate and distinguish an even more accurate "pattern of life."

5 Conclusion and Future Work

This study discussed a methodology for extracting network traffic, selecting features from Bro logs, creating time series from selected features, as well as provided results for distinguishing a "pattern of life" from a month's worth of captured network traffic. This study shared the roadblocks experienced throughout the research and the progress made in characterizing UVA's baseline network traffic. Although this characterization methodology discussed methods in distinguishing the "pattern of life" of UVA's network traffic, further work is necessary for more reliable baseline network characterization results.

While the "pattern of life" was able to be distinguished from the study's ARIMA model, there are still improvements that can be made to its overall accepted error rate. Future work should consist of expanding the current time window in order to enable this research's models to compare days, weeks, months, and years in network traffic behavior. This further expansion will aid in increasing this study's models' fitting and predictive accuracy by providing more days, weeks, months, and years of network traffic to the layered dataset. In addition, future work should aim to apply more complex models in order to train and test predictive accuracy. This will increase the ability to properly distinguish the "pattern of life" of UVA's network traffic and detect traffic not

within the accepted the error rate. Although a significant amount of noise still exists within the study's network traffic, this highlights that an anomaly would only have to be stronger than the noise accepted by the "pattern of life" to be detected. Once a significant model that can accurately characterize UVA's baseline network is developed, the next step is to inject test points into the network traffic to see how well this study's model can detect these artificial anomalies. The end result for potential future research will be to develop the most accurate model in distinguishing the "pattern of life" of UVA's network traffic in order to accurately detect network anomalies. This tool would in return aid UVA and other large academic institutions in detecting network intrusions.

Acknowledgment

The author would like to thank the following advisers and colleagues from the University of Virginia network and cybersecurity project team for their mentorship, advice, and support during this research: Dr. Donald Brown, Dr. Jack Davidson, Dr. Peter Beling, Dr. Porter, Mr. Brendan Abraham, Mr. Jack Morris, Mr. Alastair Nottingham, and Mr. Curtis Khan.

References

[1] B. Hughes, D. Bohl, M. Irfan, E. Margolese-Malin, and J. Solorzano, "Cyber Benefits and Risks: Quantitatively Understanding and Forecasting the Balance," Frederick S. Pardee Center for International Futures, Josef Korbel School of International Studies, University of Denver, Report, September 2015, pp. 23-25.

[2] J. Garamone, "Cyber Tops List of Threats to U.S., Director of National Intelligence," U.S. DEPARTMENT OF DEFENSE, February, 2018. [Online]. Available: https://www.defense.gov/News/Article/Article/1440838/cyber-tops-list-of-threats-to-us-director-of-national-intelligence-says/.

[3] "Cyberattack hits UVa," The Daily Progress, August 14, 2015. [Online]. Available: https://www.dailyprogress.com/news/cyberattack-hits-uva/article_59b0454c-42c7-11e5-88ae-53d47ac2265c.html.

[4] K. B. Williams, "Hackers hit University of Virginia," TheHill, February 4, 2016. [Online]. Available: http://thehill.com/policy/cybersecurity/251259-hackers-hit-university-of-virginia.

[5] D. Eroz, M. Yousif, and C. Das, "Characterizing network traffic in a cluster-based, multi-tier data center," 27th International Conference on Distributed Computing Systems (ICDCS'07), IEEE, 2007.

[6] D. Kotz and K. Essien, "Characterizing Usage of a Campus-wide Wireless Network," Dartmouth College, March 2002.

[7] M. Joshi and T. Aldhayni, "A Review of Network Traffic Analysis and Prediction Techniques," arXiv preprint arXiv:1507.05722, 2015.

[8] Y. Yu, M. Song, Z. Ren, and l. Song, "Network Traffic Analysis and Prediction Based on APM," 2011 6th International Conference on Pervasive Computing and Applications, IEEE, 2011.

[9] CT. Huang, S. Thareja, and Y.-J. Shin, "Wavelet-based Real Time Detection of Network Traffic Anomalies," 2006 Securecomm and Workshops, IEEE, 2006.

[10] A. A. Cardoso and FHT Vieira, "Adaptive estimation of Haar wavelet transform parameters applied to fuzzy prediction of network traffic," Signal Processing 151, 2018: 155-159.

[11] M. Price-Williams and N. Heard, "Statistical Modelling of Computer Network Traffic Event Times," arXiv preprint arXiv:1711.10416, Imperial College London, 2017.

[12] J. Gamboa, "Deep learning for time-series analysis," arXiv preprint arXiv:1701.01887, 2017.

[13] BJ Radford, L. Apolonio, A. Trias, and J. Simpson, "Network traffic anomaly detection using recurrent neural networks," arXiv preprint arXiv:1803.10769, 2018.

[14] S. Makridakis, E. Spiliotis, and V. Assimakopoulos, "Statistical and Machine Learning forecasting methods: Concerns and ways forward," PloS one 13.3, e0194889, 2018.

[15] J. Gao, H. Ling, E. Blasch, K. Pham, Z. Wang, and G. Chen, "Pattern of Life from WAMI Objects Tracking based on Context Aware Tracking and Information Network Models," SPIE, Baltimore, MD, 2013.

[16] B. Wylie, "brothon," PyPI. [Online]. Available: https://pypi.org/project/brothon/.

[17] "Bro Logs," [Online], Available: http://gauss.ececs.uc.edu/Courses/c6055/pdf/bro_log_vars.pdf.

[18] "Amazon Web Services (AWS) - Cloud Computing Services," Amazon, [Online], Available: https://aws.amazon.com/.

[19] "Advanced Research Computing Services," Rivanna — Advanced Research Computing Services. [Online]. Available: https://arcs.virginia.edu/rivanna.

[20] R. H. Shumway and D. S. Stoffer, "Time series analysis and its applications: with R examples," Cham, Switzerland: Springer, 2017.

[21] "UC Business Analytics R Programming Guide," University of Cincinnati, [Online], Available: http://uc-r.github.io/ts_exp_smoothing.

[22] "UC Business Analytics R Programming Guide," University of Cincinnati, [Online], Available: http://uc-r.github.io/ts_exp_smoothing.

[23] "UC Business Analytics R Programming Guide," University of Cincinnati, [Online], Available: http://uc-r.github.io/ts_exp_smoothing.

[24] R. Hyndman, "Forecasting with long seasonal periods," 2005, [Online], Available: https://robjhyndman.com/hyndsight/longseasonality/.

Appendix

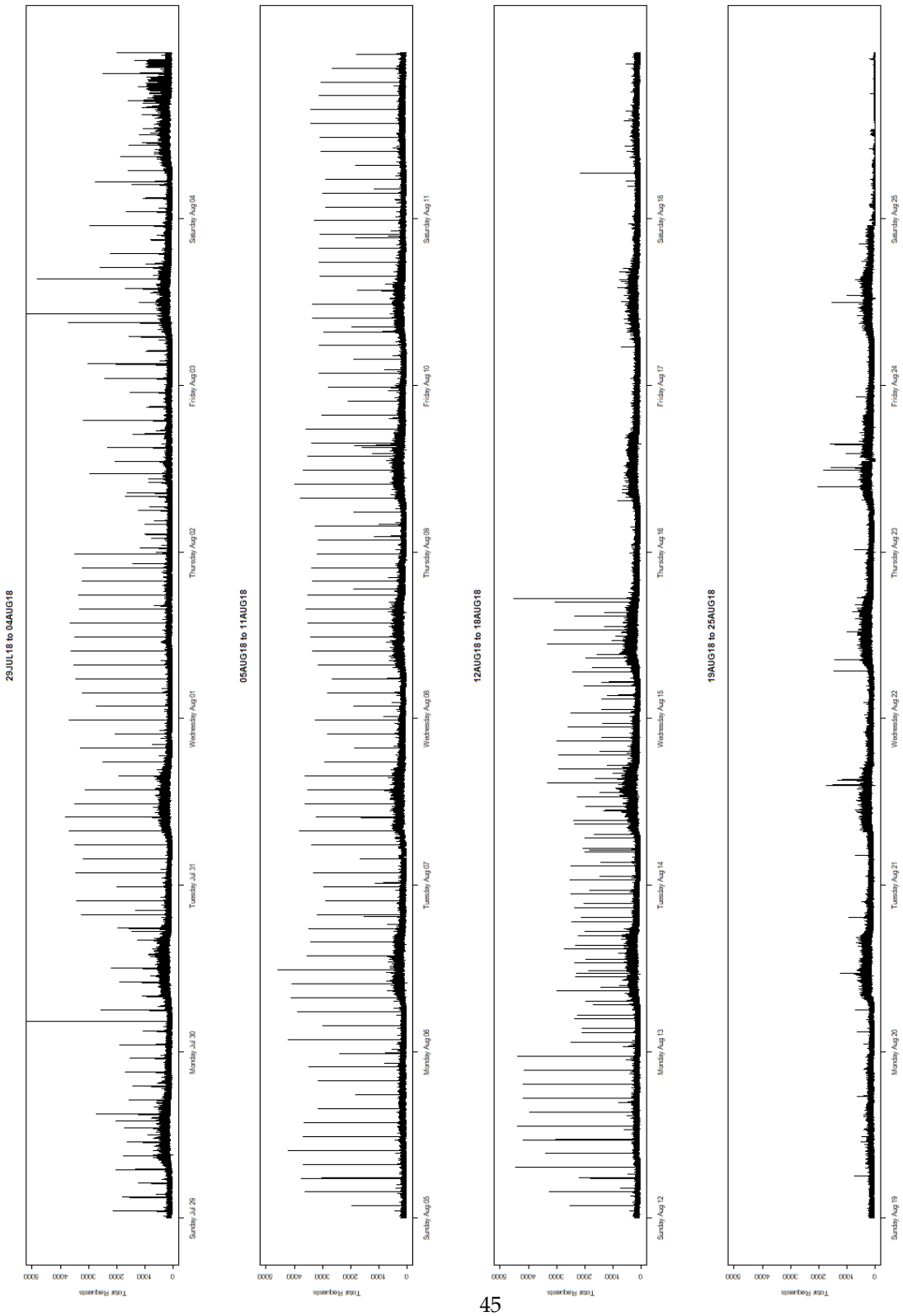

Figure 14: Time Series Graph for comparing each week of traffic between July 29, 2018 and August 25, 2018.

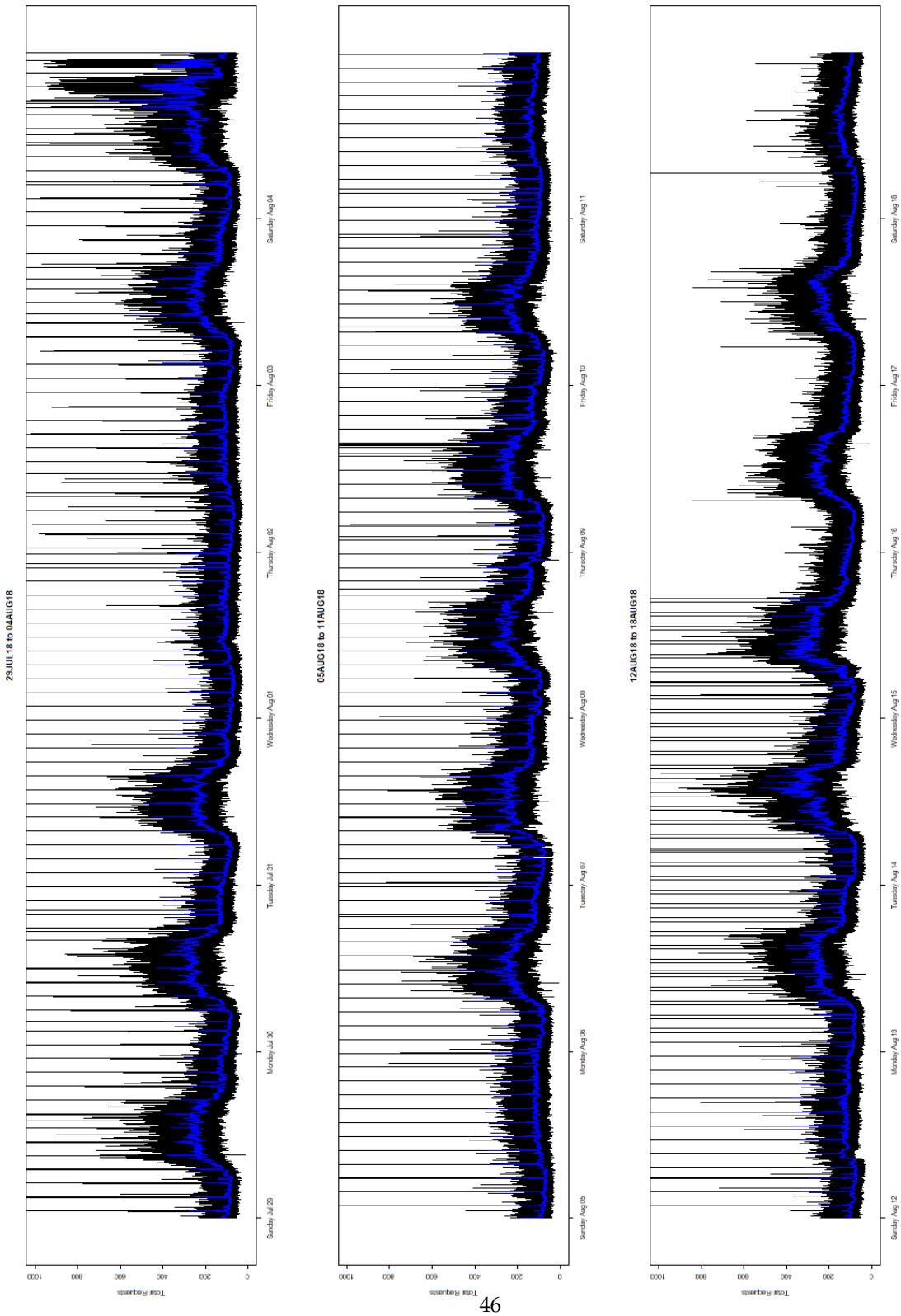

Figure 15: Time Series Graph for displaying fitting of Exponential Smoothing Model.

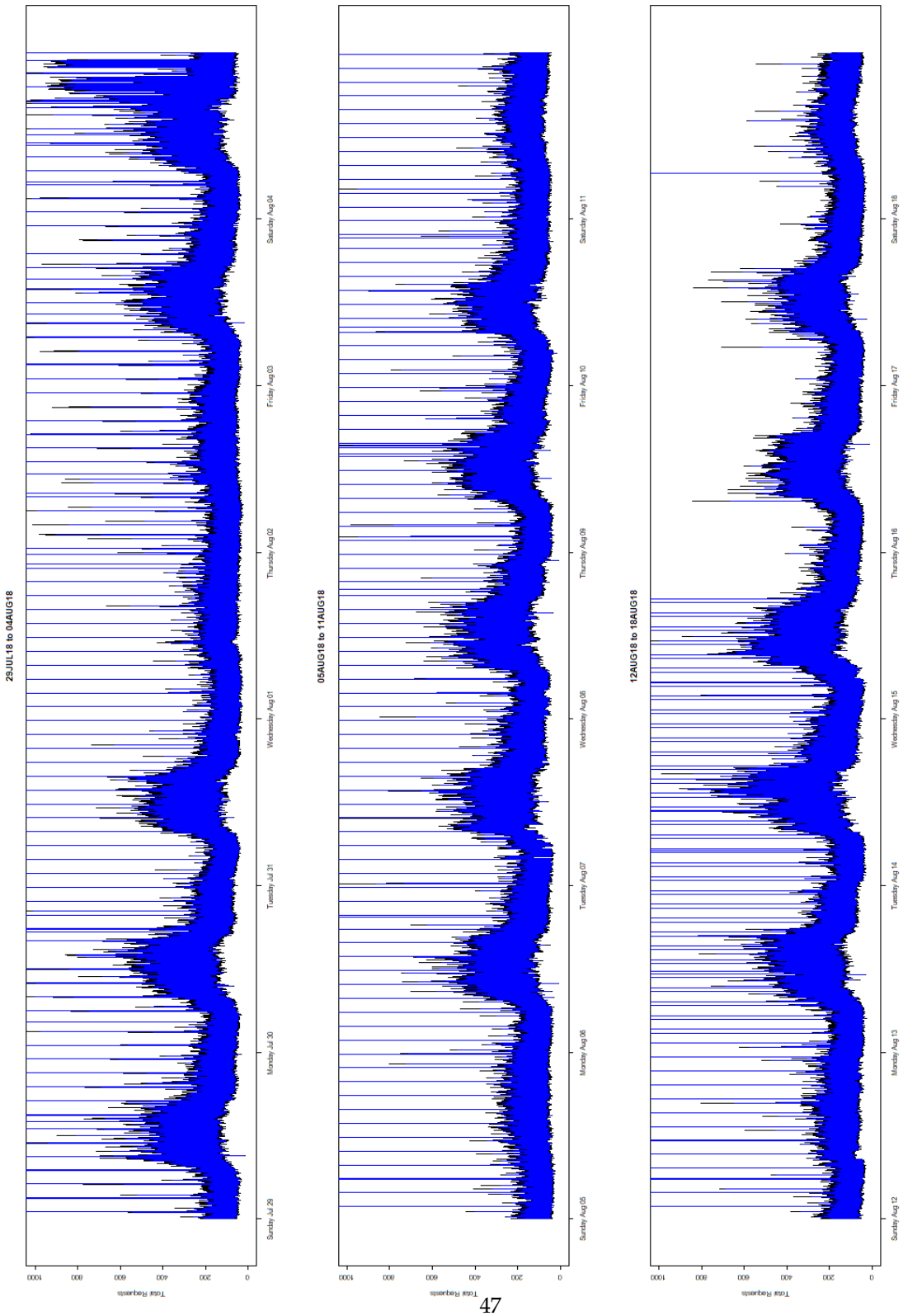

Figure 16: Time Series Graph for displaying fitting of Additive Model

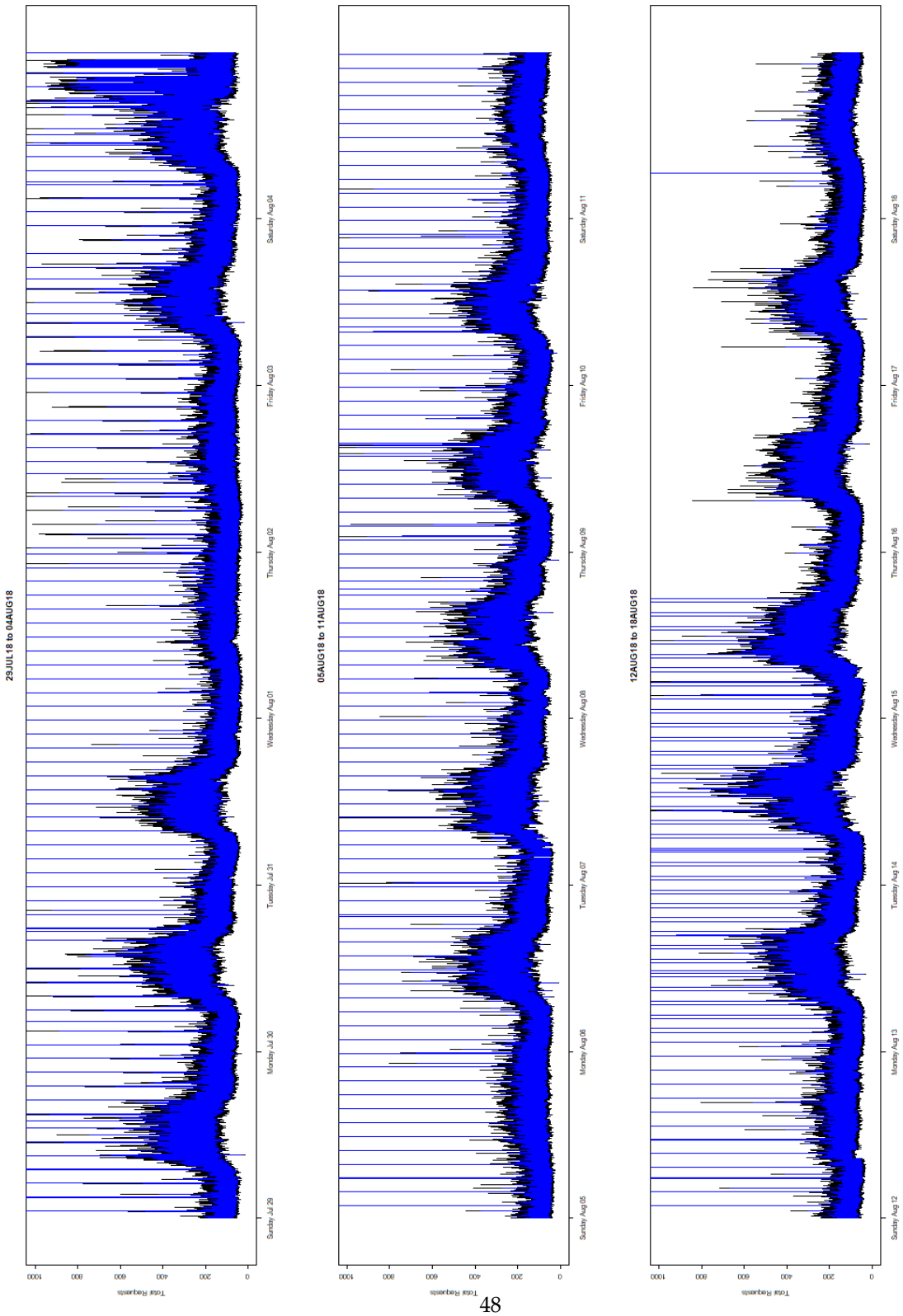

Figure 17: Time Series Graph for displaying fitting of Multiplicative Model

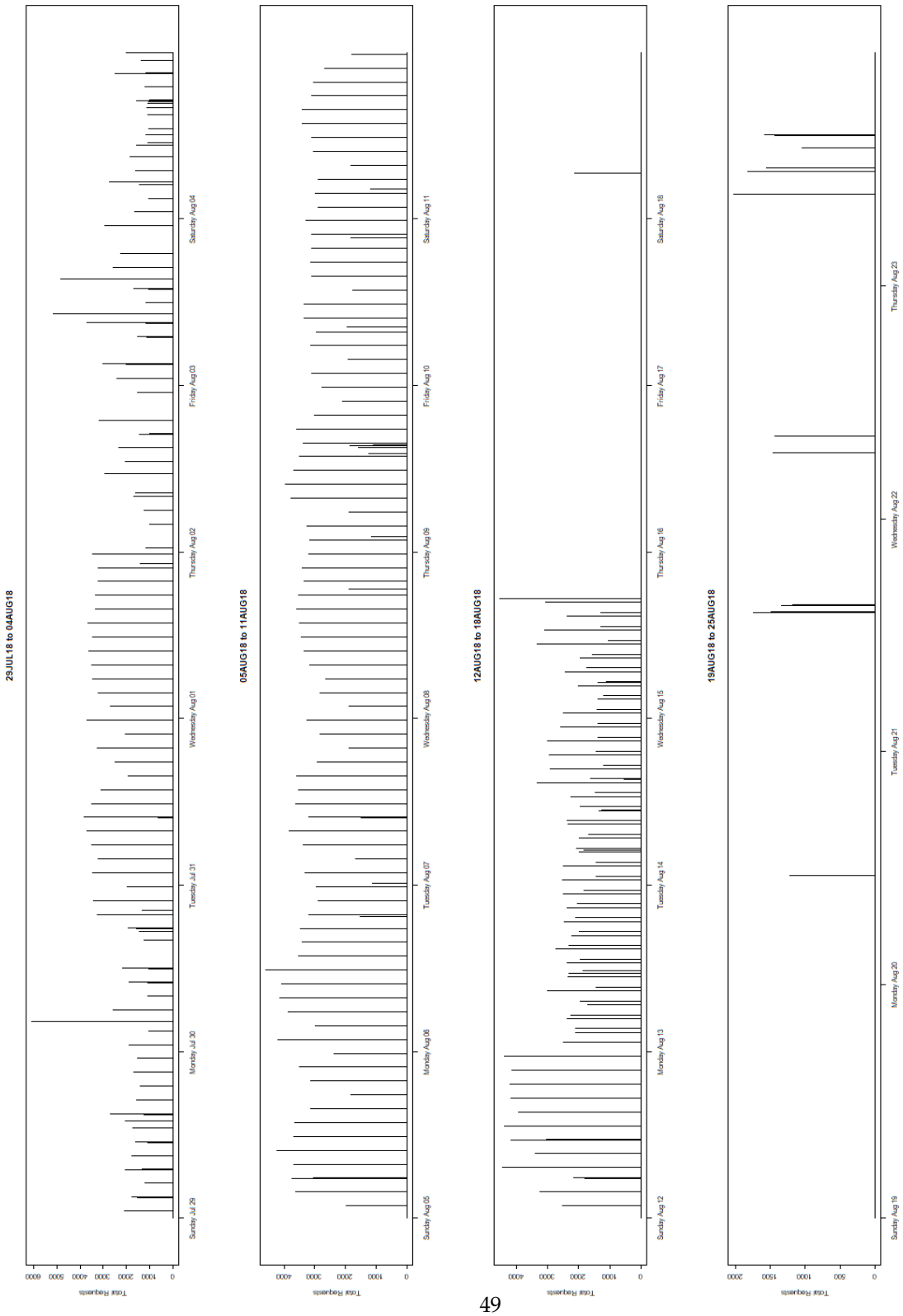

Figure 18: Time Series Graph for displaying each outlying spike between July 29, 2018 and August 25, 2018.

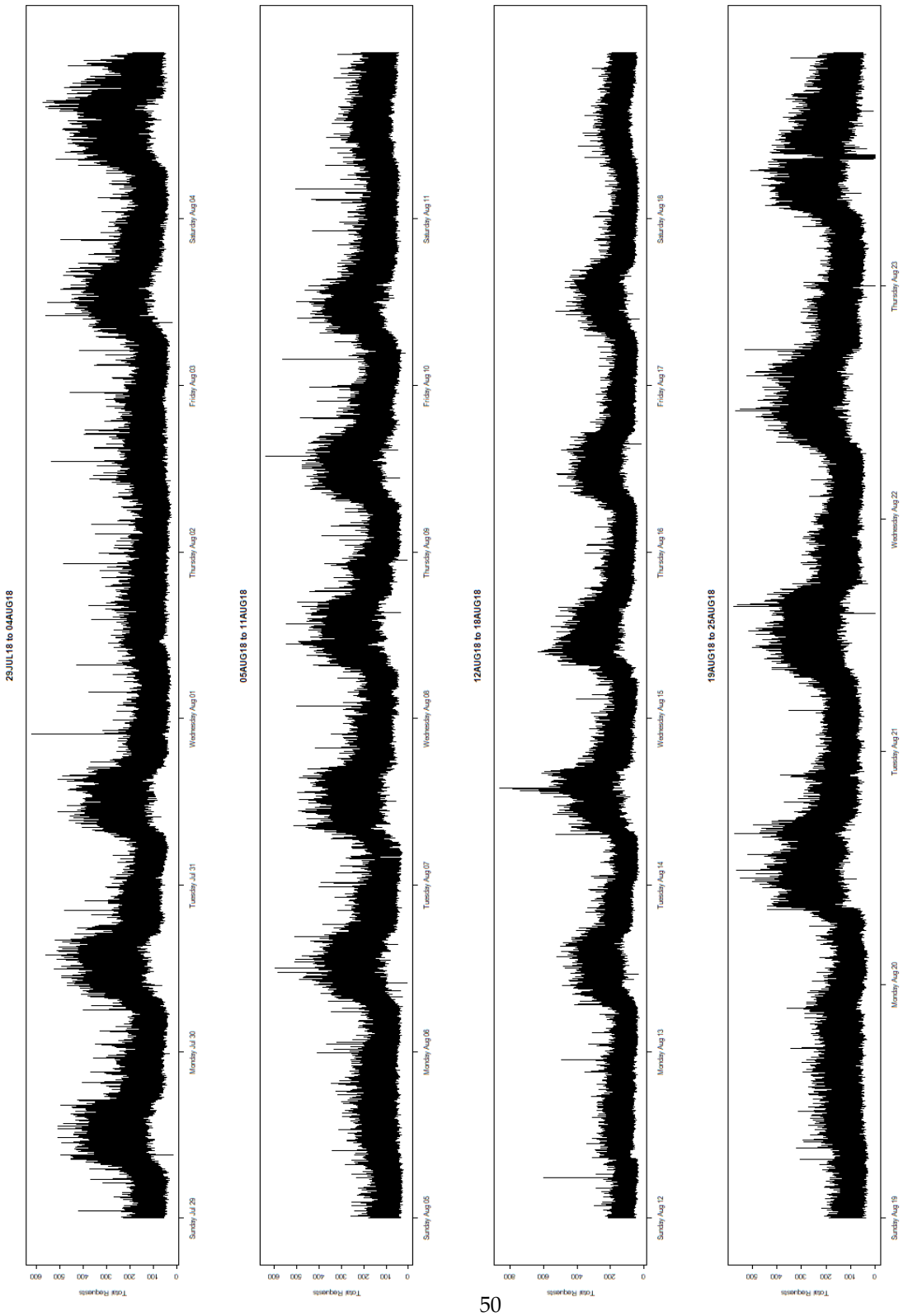

Figure 19: Time Series Graph for displaying results of imputation between July 29, 2018 and August 25, 2018.

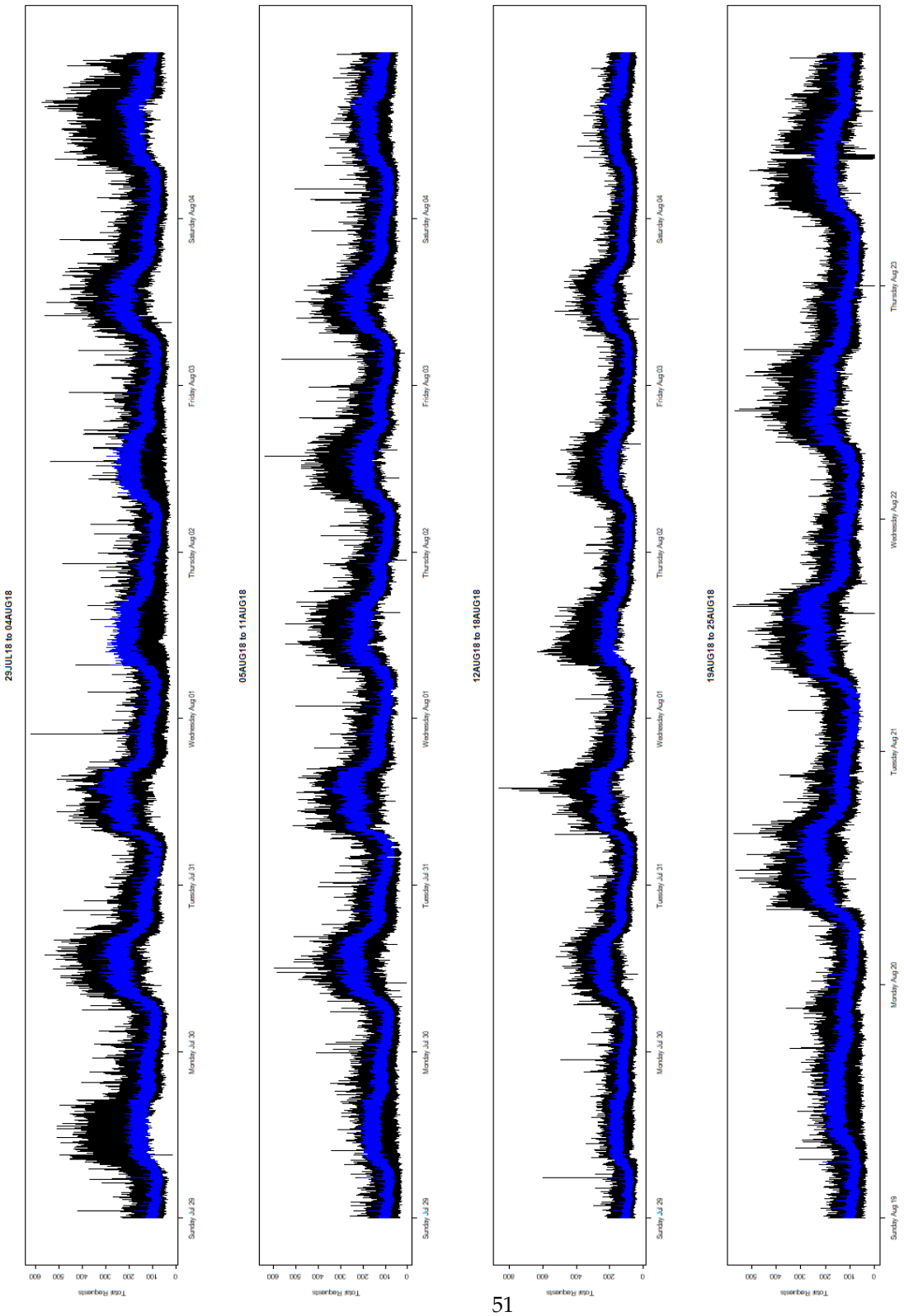

Figure 20: Time Series Graph for displaying results of ARIMA Baseline Model with imputation between July 29, 2018 and August 25, 2018.