Feedback Control for Resiliency in Face of an Attack

Jagannadh Vempati
Department of Computer Science and Engineering
University of North Texas
Denton, USA
jagannadhvempati@my.unt.edu

Mark Thompson
Department of Computer Science and Engineering
University of North Texas
Denton, USA
mark.thompson2@unt.edu

Ram Dantu
Department of Computer Science and Engineering
University of North Texas
Denton, USA
rdantu@unt.edu

ABSTRACT

Distributed Denial of Service(DDoS) attacks are inevitable. The existing defensive mechanisms are relatively outdated. In this paper, we present a passive mechanism to reduce the impact of an attack on the network. We designed and implemented a robust feedback architecture, to maintain the stability of the network despite attacks. During an attack, the controller of the feedback architecture detects the irregularities in the response time and the necessary changes are made to the configuration to maintain the network in steady state. In this approach first, we model the network using black-box system identification technique. Second, we validate the model with test data by conducting various experiments such as varying the network topology. Last, we test the model with the feedback architecture built in our lab environment. Results show that the feedback architecture provides an average model fit accuracy with positive results.[1]

CCS CONCEPTS

• **Control system theory** → **Feedback control**; *System Identification* • **Networks** → Network resilience; *Distributed Denial of Service attack (DDoS)*

1 INTRODUCTION

On October 21, 2016, Dyn, Inc., an Internet infrastructure company that provides core Internet services for Twitter, Reddit, and Spotify among others, sustained a distributed denial of service (DDoS) attack that took down a significant portion of the Internet on the U.S. East Coast [1]. Despite their best efforts, the effects of the multiple attacks were felt for several hours as Dyn used a wide range of techniques in their armor such as traffic shaping and rebalancing, internal filtering, and scrubbing services to recover from the attacks [2]. One problem with a DDoS attack such as this is that it may be difficult to distinguish between legitimate traffic and attack traffic. In fact, these attacks generated a barrage of legitimate retry activity that only made the impact on their customers and end users even worse.

Interest in network security is skyrocketing due to the numerous highly visible and severely crippling attacks of our computers and networks and the resulting perception that our digital presence and environment is no longer secure. Despite the existence of a number of cybersecurity tools and techniques, many networks are still vulnerable to malicious attacks. Since network systems are complex, unpredictable, and highly dynamic, static techniques that are reactive in nature are not able to effectively respond to the changing environment in real-time. In this manner, feedback control systems can play a vital role in mitigating these malicious attacks in real-time [21] by detecting and controlling the connections made by an infected host.

Understanding the dynamic nature of traffic in a network is critical to being able to create and maintain an accurate model. Continuous revision and understanding of the model for a variety of given inputs and outputs, therefore, is critical to the success of being able to provide resiliency in the face of an attack or failure in the network. The response of the network as well as service disruptions should also be studied carefully to be able to characterize the behavior of the system. Once these components are understood, feedback mechanisms can then be applied to allow the network to adapt its configuration in response to an attack or failure to provide and maintain an acceptable level of throughput.

Current mitigation techniques are extremely slow and subject to error, especially in the face of changing attack vectors that can vary in size and approach. In this paper, we propose a robust feedback mechanism to provide resilience in a network that is both scalable and effective in real-time. We look to system identification to design a model of our complex and dynamic network. We then validate our model using a configurable network that can act upon feedback received from our transfer function model and respond to disruptions in the network. The importance of this mechanism is that it provides a passive, real-time, and scalable solution to mitigate the impact of an attack on the network.

1.1 RELATED WORK

Control theory approach has been applied to wide range of computing systems [7]. There has been significant amount of research on identifying and performance modeling of web servers [5, 13, 15, 17]. Most of these models are based on linear time invariant (LTI). There have been also several approaches for designing software systems using control theory [14]. An LPV approach to performance modeling of web server on a private cloud was presented in [12]. In this work an operating region for

CISRC '17, April 04-06, 2017, Oak Ridge, TN, USA
© 2017 ACM. ISBN 978-1-4503-4855-3/17/04…$15.00
DOI: http://dx.doi.org/10.1145/3064814.3064815

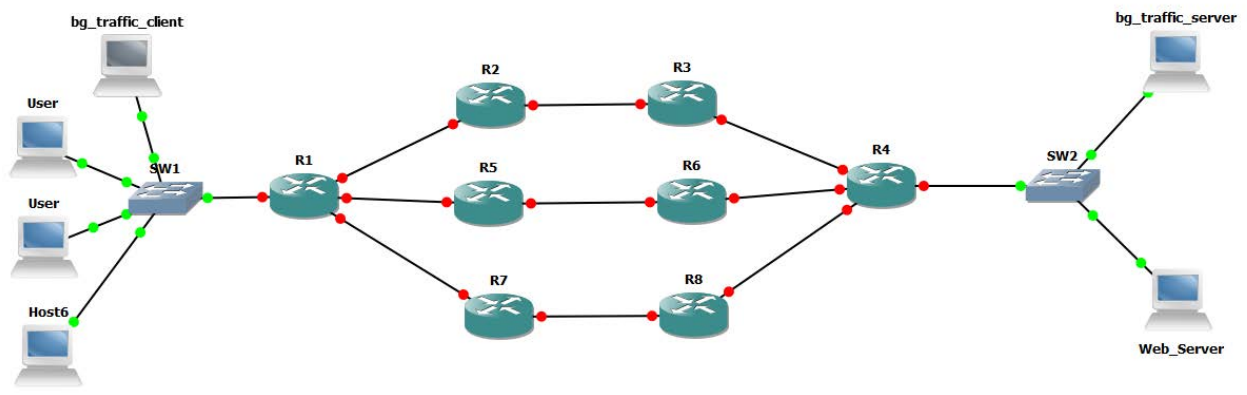

Figure 2: *Experimental Network Topology: Network Elements Connected in Parallel*

the web server was defined around which the LPV state space model was derived. An LPV approximation for admission control of a web server was presented in [15]. T Patikirikoral et al., proposed a Hammerstein-Weiner nonlinear model based control for quality of service(QoS) management [18]. A control oriented model for performance management in virtual environment was suggested by Dharma Aryani et al., [19]. Most of these works concentrate on performance analysis of a web server and software systems. An Architecture for network security using feedback control has been proposed in [21]. These mechanisms do not model the attack.

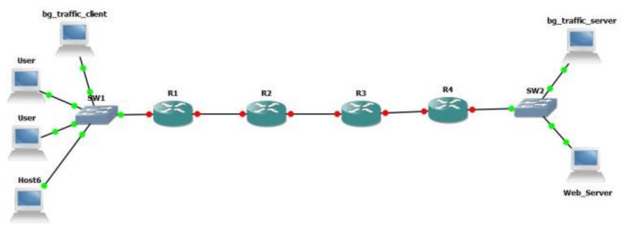

Figure 1: *Experimental Topology: Network elements connected in series*

To the best of our knowledge, the use of feedback control to build resilient network has to be yet explored. However, there has been work done on the resilience problem of routing in a parallel link network with a malicious player using game theoretic approach [22]. But their focus was on building efficient solution algorithms. The solution was a theoretical approach.

Our approach has a solid defense mechanism. The feedback mechanism implemented in our approach stabilizes the network and makes the network function robustly despite of attack. Also, we prove that the QoS of the network remains same even after the network is disrupted.

2 ARCHITECTURE

2.1 Network Topology

To demonstrate the effectiveness of our feedback mechanism, we implement a network connected in parallel using eight Cisco Catalyst devices as shown in Figure 2. The network itself consists of three parallel links, each comprising of four routers connected in series. Each router is a configured to use the Open Shortest Path First (OSPF) protocol. The three-parallel links are configured such that they share the load equally. The administrative distances between the routers are made identical. The router R4 in Figure 2 is configured to perform the load-balance operation per destination. This load-balance feature in the Cisco router distributes the packets based on the destination address.

The design of this network is deliberate so as to be able to measure the performance of our network relative to one connected in series as shown in Figure 1. In this manner, we are able to study the response of the network and establish benchmarks needed to characterize the behavior of the system and ultimately understand the dynamics of the system and be able to respond to disruptions or failures in the network.

2.2 Client

Linux containers (LXC) [11] were used to emulate real word scenario. LXC is an operating system level virtualization that is able to run multiple Linux hosts on a single system. These containers can resemble clients in sending requests to a web server. These containers are connected through a bridge interface in a same subnet. In this manner, we are able to model a significant number of users making requests to the server.

2.3 Server

A standalone machine is used to host an Apache HTTP web server [10] using an Intel Xenon processor (4 cores) with 32 GB RAM. The Apache server is configured to accept 50,000 requests per second. This configuration is required to ensure that all of the client requests in our model are served without being rejected. The Keep-Alive directive is enabled in the configuration to reduce the number of connections that will ensure the server does not become overloaded.

3 METHODOLOGY

3.1 System Identification

To be able to analyze the stability and sustainability of a network so as to design a robust feedback system, a system model representing the complex network is required. A lower-order linear model is not adequate to model the dynamics of such an intricate network. We are thus motivated to derive a model using the system identification technique. System identification is the science of building analytical models of dynamic systems from observed input-output data [4]. Using this process, we are able to obtain a mathematical relation between the input and output data.

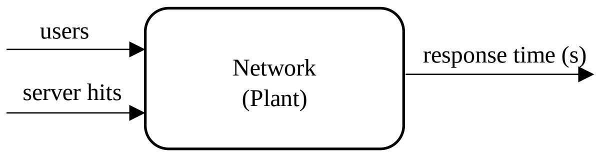

***Figure 3**: Open loop system. The input variabes are number of users and server hits; The output variable measured is response time to access the web page.*

We use a black box approach to identify the model of the plant as shown in Figure 3. We consider the complete network comprising of clients, network devices, such as routers, switches, etc., and web server as a plant. We then model this plant using the system identification toolbox present in Matlab [8]. The input parameters to the plant are the number of users browsing the web server and the number hits, or requests, per second to the web server. The output parameter that is measured is response time, which serves as a Quality of Service (QoS) metric.

3.2 Assumptions

The following assumptions are made to identify and model the system:

1) We consider two inputs: the number of users making requests of the web server and the number of hits per second. We define the number of hits as the number of new or unique users sending requests to the server.
2) We limit the total number of users to 20,000.
3) The web server serves a single static page with a size of 200 KB.
4) We consider one output: the response time.
5) We consider the system to be stable when connected in parallel and modeled the system accordingly.

3.3 Generation of Client Requests

We use the Apache Jmeter load tester to generate the HTTPS requests as well as to measure the performance of the server in terms of the response time. In particular, we apply the distributed testing [23] component of Jmeter to send multiple HTTPS requests from the Linux containers. In distributed testing, one master node, or controller, initiates the test on multiple slave systems. The Linux containers, operating as slave systems, can then periodically send the QoS performance data such as response time, latency, number of requests, etc. to the master controller. The master controller also collects the performance metrics from the server, such as CPU and memory utilization, thus acting as a sensor that records the output data periodically.

3.4 Background Traffic

We use the hping3 [3] tool as a background traffic generator to insert homogenous traffic along the routers R1 and R3 as shown in Figure 2. This tool can generate this background traffic when the network is connected in series or when the parallel links are enabled, distributing traffic evenly among the three active paths.

3.5 Feedback

Prior to being able to implement the feedback architecture shown in Figure 4, we identify the applicable model using the system identification toolbox. We then conduct several experiments to determine the region of stability, or acceptance, for the response time and thus are able to identify the dynamic characteristics of the system accordingly. In doing so, we found that the network is more resilient when connected in parallel compared to the network connected in series Section 3.1. Hence, we consider this parallel network as a benchmark for our model.

With the network initially connected in series, the feedback to the network is applied by enabling the parallel links. The closed loop feedback model of our approach is shown in Figure 4. For the purposes of this paper, we consider the homogenous background traffic to be legitimate traffic. The response time is fed back to the proportionality controller, which is aware of the network topology. This feedback is then used to detect anomalies in the response time. When the network is attacked, the network can go into an unstable state. With the help of the feedback mechanism, the controller detects the irregularities and disturbances of the response time so that the necessary configuration changes can be made to bring the network back to the desired state.

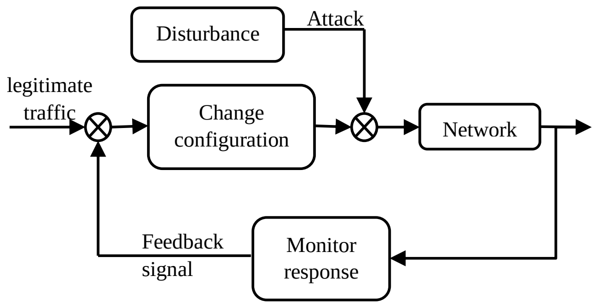

***Figure 4**: Automatic Feedback Control: Real-time Configuration Changes in the event of a Disturbance (e.g., DoS attack)*

4 ANALYSIS OF RESULTS

Our approach is to initially model traffic when the network connectivity is fixed in series and in parallel to establish a baseline of what we would expect our input-output traffic profile to look like

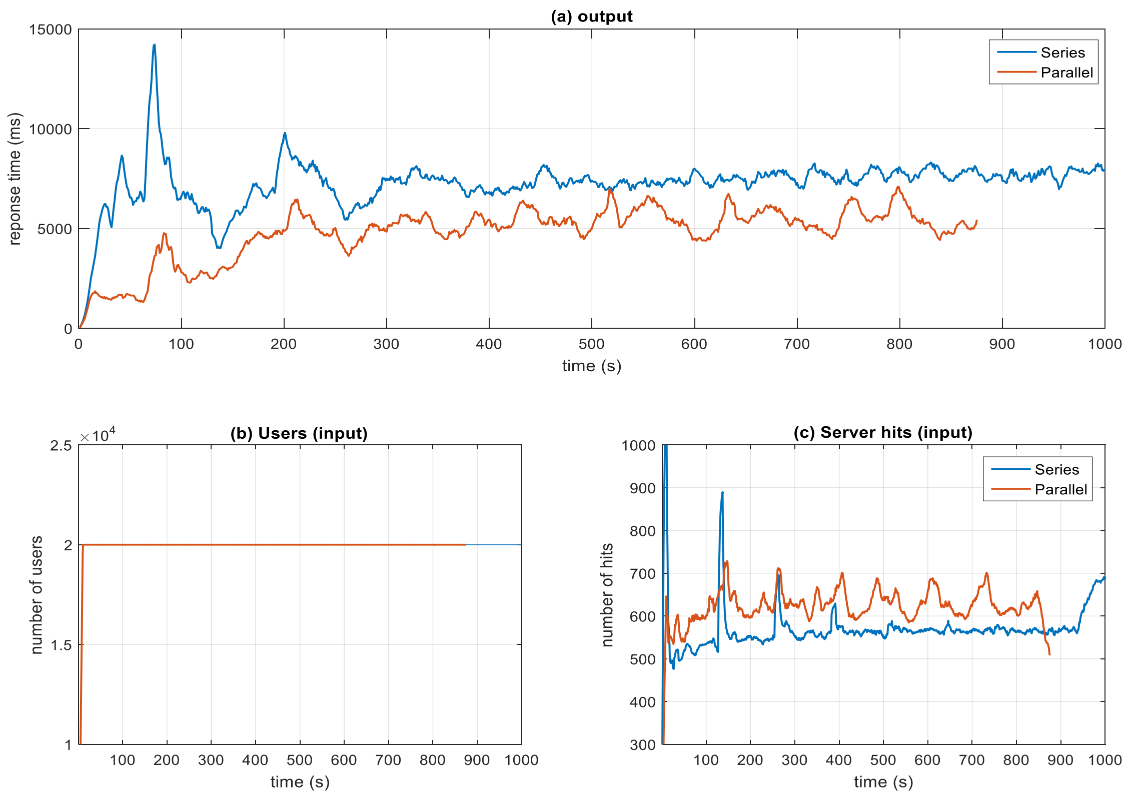

Figure 5: *System response for the network connected in series and in parallel topologies. (a) Response time for the HTTPS requests for the series and parallel network configuration (b) and (c) are the number of users and number of hits connected in series and parallel.*

during an attack. Figure 5 shows this input-output of attack traffic with background traffic data collected from the network connected in series and in parallel. The input to the network is the number of users and the hits to the server. The output of the network is the corresponding response time, measured in milliseconds.

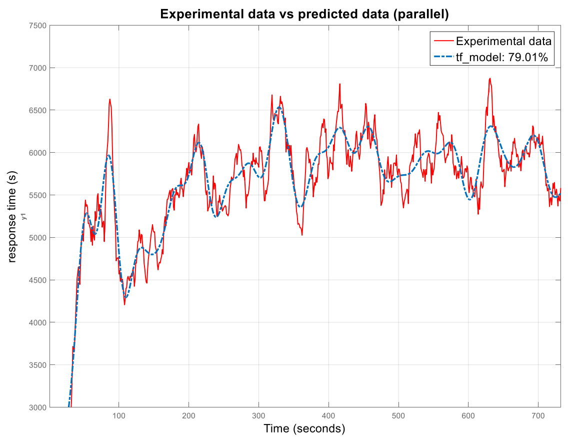

Figure 6: *The transfer function (through system identification) closely follows the experimental data even during an attack. (observed 79% match). Parallel network topology was used as shown in Figure 1,*

As is shown, the network is loaded with a step input of 20,000 users sending requests to the web server. We can easily observe that, when given the same number of users making requests to the server, the response time of the network performed significantly better than that of the network connected in series, as we expect, resulting in approximately a 25% drop in the response time. As a result of this increase in throughput, the number of successful hits, or requests, increased by approximately 16% for the network connected in parallel. Thus, we would want our feedback system to be able to bring our input-output traffic profile, when connected in series that of the traffic profile for the network connected in parallel.

4.1 Stability analysis

4.1.1 Stable system. Here, we focus on the stability of networks. In particular, we note that network traffic itself can be unstable, perhaps due to large fluctuations in demand brought about by a varying number of users and retry attempts in the case of timeouts or other message failures. First, we attempt to model the steady state, and then compare the responses with the model using fit accuracy in such that the higher the fit, the closer, or more stable, our data is to the model. Figure 7 shows the fit accuracy of the predicted transfer function model against the traffic data collected from the network connected in series using only attack traffic.

Since our model shows a fit accuracy of approximately 81%, we establish this transfer function model as a baseline for a stable region.

We then model the steady state for attack and background traffic when the network is connected in parallel, as is seen in Figure 6. We observe that the fit accuracy for our network connected in parallel is nearly 80%, which now serves as our benchmark for us to attain for our network under attack.

Table 1 summarizes the average response time in table form when the network is connected in series as well as in parallel. We observe that the average response time in steady state for networks connected in parallel is similar to networks connected in series up to 5,000 users. As the number of users increase, however, the response time of the network connected in series tends to diverge from that of the network connected in parallel.

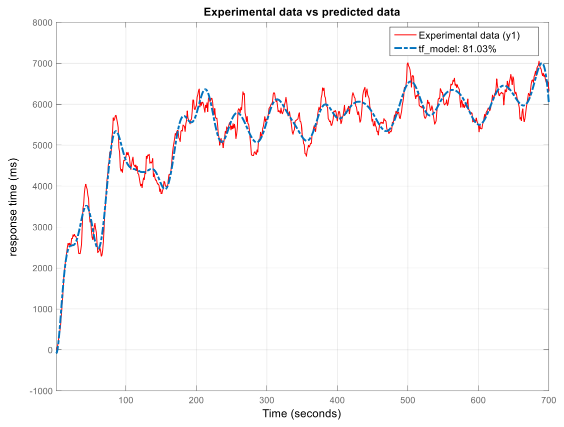

Figure 7: *The identified transfer function model closely follows the data collected from the network under attack connected in series without the presence of background traffic.*

4.1.1 Unstable system. Although we observed that when we ran attack traffic only in our network connected in series, we obtained a rather stable system where our transfer function model yielded a very good fit. In Figure 8, we now see that when background traffic is added to our network running in series, the fit accuracy of our model drops significantly to approximately 33%, indicating that our system is now unstable, most likely because the nodes are congested possibly due to an attack.

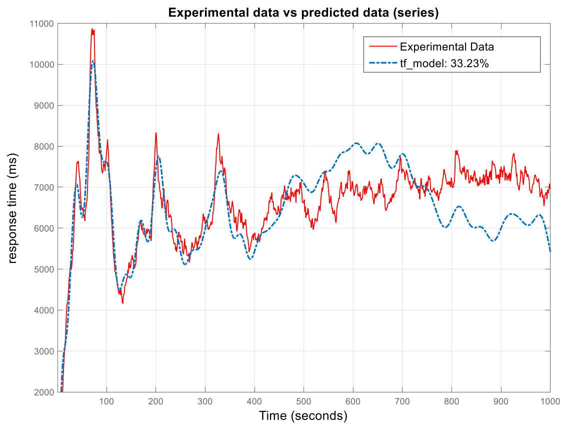

Figure 8: *The experimental data does not follow the identified transfer function when the network is under attack. This poor fit % indicates that the network is unstable.*

4.2 Feedback Application

We now show our input-output traffic profile with attack and background traffic when feedback is applied to our network connected in series once the system becomes unstable, causing the network connection configuration to change from series to parallel. When connected in parallel, the load is shared among the links, causing in a decrease in the response time for client requests. As a result, the number of hits increases as expected because of the increase in throughput. Table 2 shows the accuracy of the predicted data against the experimental data when the network is connected in both series and parallel. It is important to note that as the number of users increases in a network connected in series, the fit accuracy of our model drops significantly once the number of users reaches 5,000 or more. The model accuracy for networks connected in parallel, however, remain steady throughout the attack.

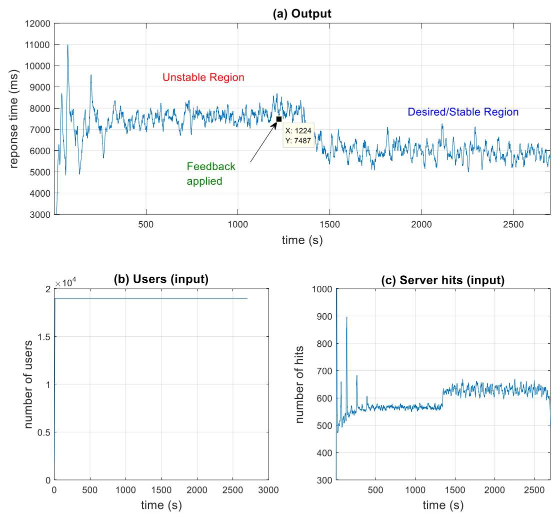

Figure 9: *(a) Response time of the network before and after the attack . (b) and (c) are the inputs provided to the network. The feedback is applied at time = 1224 seconds to the network under attack. The network goes to stable region from unstable state within 300 seconds. (c) shows the increased number of hits due to the increase in throughput.*

What is important in Figure 9 is that once our model detects instability of the network, our feedback mechanism has a positive effect on the network in real-time, not in hours as we saw in the case of the DNS attack on Dyn that resulted in a significant amount of downtime for its customers and end users. Figure 10 shows the stable region of Figure 9 after the feedback mechanism was applied that jumps back up to have a fit accuracy of approximately 75%, meaning that the network is back in a stable state. Although we did not actively stop the attack, we were able to use a passive approach to in fact mitigate the attack.

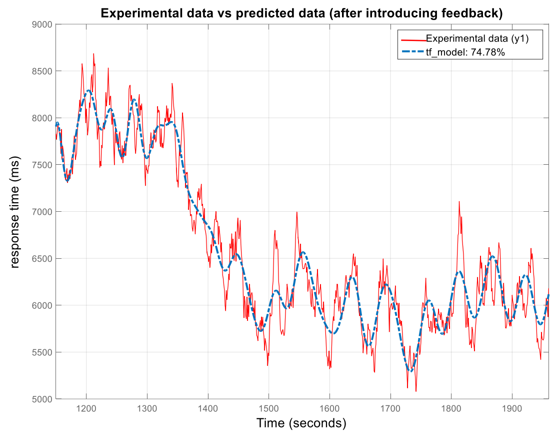

***Figure 10:** Extension of Figure 9; The transfer function model closely follows the data after the feedback is introduced. This proves that the network goes to stable state from unstable state after feedback is applied.*

Table 1: *This table summarizes the average response time, when the network is connected both in series and parallel*

No. of users	Average response time (ms)	
	Series	Parallel
500	366	272
1000	1000	900
5000	3600	3500
10000	4800	3800
15000	6600	4300
20000	7800	6000

Table 2: *This table summarizes the average response time, when the network is connected both in series and parallel*

Model Accuracy (%)		
No. of users	Series	Parallel
1000	64.91	64.34
5000	70.36	66.32
10000	74.62	42.23
15000	76.31	41.11
20000	80	33.25

5 CONCLUSION AND FUTURE WORK

The attack traffic is modeled in this approach to ensure the network maintains a stable state with any type of legitimate traffic flowing through the network. Table 1 shows that the passive mechanism adapted proves to be promising in maintaining a steady state of the network even when it is disrupted. We used a proportionality controller for this paper, though we suspect significantly better results may be obtained by using a PID and a PI controller.

Our model is trying to fit all the data points in a higher order system, which may yield a slightly lower percentage of accuracy. The model can be fine-tuned to a lower order for a better fit accuracy. When the data is filtered further, lower order models yield better accuracy; however, we might lose critical data.

In the future, we will fine-tune the model by building a better controller. The response time does not show a drastic drop regardless of adding three parallel links. This is due to the bottleneck added at the server. This issue should be able to be resolved using a distributed server architecture. We would also like to extend this architecture to distributed servers to load-balance the requests. We would like to test this feedback mechanism in Software Defined Networks architecture where control theory proves to be more effective

REFERENCES

[1] Chacos, Brad. "Major DDoS Attack on Dyn DNS Knocks Spotify, Twitter, Github, PayPal, and More Offline." PCWorld. PCWorld, 21 Oct. 2016. Web. 07 Dec. 2016., http://www.pcworld.com/article/3133847/internet/ddos-attack-on-dyn-knocks-spotify-twitter-github-etsy-and-more-offline.html
[2] Hilton, Scott. "Dyn Analysis Summary Of Friday October 21 Attack." Dyn Blog. Dyn News, 26 Oct. 2016. Web. 8 Dec. 2016., http://dyn.com/blog/dyn-analysis-summary-of-friday-october-21-attack/
[3] http://www.hping.org/hping3.html
[4] L. Ljung. System Identification: Theory for the User. 2nd ed. Prentice Hall, Upper Saddle River, NJ, 1999.
[5] J. Doyle, B. Francis, and A. Tannenbaum. Feedback control theory. MacMillan, 1992
[6] T. Abdelzaher, Y. Diao, J. Hellerstein, C. Lu, and X. Zhu. Introduction to control theory and its application to computing systems. In Performance Modeling and Engineering, pages 185–215. Springer US, 2008
[7] J. L. Hellerstein, Y. Diao, S. Parekh, and D. M. Tilbury. Feedback Control of Computing Systems. John Wiley & Sons, 2004
[8] http://www.mathworks.com/products/sysid/
[9] http://jmeter.apache.org/index.html
[10] https://httpd.apache.org/
[11] https://linuxcontainers.org/
[12] Saikrishna, Penamakuri Sesha, Ramkrishna Pasumarthy, and Nirav P. Bhatt. "Identification and Multivariable Gain-Scheduling Control for Cloud Computing Systems."
[13] Sun, Qilu, Guanzhong Dai, and Wenping Pan. "LPV Model and Its Application in Web Server Performance Control." In Proceedings of the 2008 International Conference on Computer Science and Software Engineering-Volume 03, pp. 486-489. IEEE Computer Society, 2008
[14] Filieri, Antonio, Martina Maggio, Konstantinos Angelopoulos, Nicolás D'Ippolito, Ilias Gerostathopoulos, Andreas Berndt Hempel, Henry Hoffmann et al. "Software Engineering Meets Control Theory."
[15] Parakh, S., Gandhi, N., Hellerstein, J. L., Tilbury, D. M., & Bigus, J. P. (2001). Using control theory to achieve service level objectives in performance management.Proceedings of IEEE/IFIP symposium on integrated network management (pp. 841–854).
[16] Qin, Wubi, and Qian Wang. "An {LPV} Approximation for Admission Control of an Internet Web Server: Identification and Control." Control Engineering Practice 15, no. 12 (2007): 1457-1467.
[17] Robertsson, A., Wittenmark, B., Kihl, M., & Andersson, M. (2004). Design and evaluation of load control in web server systems. Proceedings of American control conference (pp. 1980–1985). Boston, MA
[18] Patikirikoral, Tharindu, L. Wang, and A. Colman. "Hammerstein-Weiner Nonlinear Model Based Predictive Control For QoS Management in Complex Software Systems." Control Engineering Practice 20, no. 1 (2012): 49-61

[19] Aryani, Dharma, Liuping Wang, and Tharindu Patikirikorala. "Control Oriented System Identification for Performance Management in Virtualized Software System." IFAC Proceedings Volumes 47, no. 3 (2014): 4122-4127

[20] Symantec, "Internet Security Threat Report", 2016. https://www.symantec.com/content/dam/symantec/docs/reports/istr-21-2016-en.pdf

[21] Dantu, Ram, Joao W. Cangussu, and Sudeep Patwardhan. "Fast Worm Containment Using Feedback Control." IEEE Transactions on Dependable and Secure Computing 4, no. 2 (2007): 119-136

[22] Altman, Eitan, Aniruddha Singhal, Corinne Touati, and Jie Li. "Resilience of Routing in Parallel Link Networks." In International Conference on Decision and Game Theory for Security, pp. 3-17. Springer International Publishing, 2016

[23] http://jmeter.apache.org/usermanual/remote-test.html