A CONTROL THEORETIC APPROACH FOR RESILIENT NETWORK SERVICES

Jagannadh Ambareesh Vempati, B.Tech., M.S.

Dissertation Prepared for the Degree of

DOCTOR OF PHILOSOPHY

UNIVERSITY OF NORTH TEXAS

December 2018

APPROVED:

Ram Dantu, Major Professor
Robert Akl, Committee Member
Janice Hauge, Committee Member
Mark Thompson, Committee Member
Barrett R. Bryant, Chair of the Computer Science and Engineering Department
Yan Huang, Interim Dean of the College of Engineering
Victor Prybutok, Dean of the Toulouse Graduate School

Vempati, Jagannadh Ambareesh. *A Control Theoretic Approach for Resilient Network Services*. Doctor of Philosophy (Computer Science and Engineering), December 2018, 134 pp., 3 tables, 62 figures, 4 appendices, 117 numbered references.

Resilient networks have the ability to provide the desired level of service, despite challenges such as malicious attacks and misconfigurations. The primary goal of this dissertation is to be able to provide uninterrupted network services in the face of an attack or any failures. This dissertation attempts to apply control system theory techniques with a focus on system identification and closed-loop feedback control. It explores the benefits of system identification technique in designing and validating the model for the complex and dynamic networks. Further, this dissertation focuses on designing robust feedback control mechanisms that are both scalable and effective in real-time. It focuses on employing dynamic and predictive control approaches to reduce the impact of an attack on network services. The closed-loop feedback control mechanisms tackle this issue by degrading the network services gracefully to an acceptable level and then stabilizing the network in real-time (less than 50 seconds). Employing these feedback mechanisms also provide the ability to automatically configure the settings such that the QoS metrics of the network is consistent with those specified in the service level agreements.

Copyright 2018

by

Jagannadh Ambareesh Vempati

ACKNOWLEDGMENTS

Pursuing my Doctoral degree has been one the most fruitful achievements of my life. For this, I would like to thank my advisor and mentor Dr. Ram Dantu, who provided me with the opportunity to work in the area of cybersecurity. I am grateful to him for providing constant support, guidance, and motivation in shaping my research ideas and career. I would also like to thank my committee members Dr. Robert Akl, Dr. Janice Hauge, and Dr. Mark Thompson for their valuable time and advice to improve my research. I would also like to thank all the professors who taught me at UNT and helped me in achieving this milestone.

This dissertation would not have been possible without the blessings and encouragement of my parents, Mr. V.V.K Sastry and Mrs. V. Jayalakshmi and my brother, Mr. V.R.S Anurag. They implanted in me a dream of becoming a research scientist to make a valuable contribution to the society. I would like to dedicate this dissertation to my parents and my brother, for their unconditional love and endless belief in my abilities. Thank you, Amma, Nanna, and Chotu. I love you all.

I would also like to thank my lab mates and my friends (too many to mention!) who have been through the trials and tribulations together with me, thank you so much for being the beautiful companions and making UNT such a memorable place to study. I love you all.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Internet - a place where dreams can turn into reality and reality can feel like a dream. Every day billions of gigabytes make their way on to the Internet. Humans, being social creatures post millions of pictures, videos and also share their views and opinions for others to see. Often, this data might include deeply personal information and forms a digital footprint which is unique to each person. Data on the Internet may include memories, dreams, and thoughts. It is easy to see how digital manifestations of human lives now reside on the Internet. The sensitive nature of the data makes it a target for the people with malicious intent (threat actors).

What motivates these attackers? Hackers may impersonate other users to elicit information they otherwise wouldn't be privy to. They steal sensitive information from various prominent organizations and sell them to their competitors. State-sponsored actors, acting on behalf of governments engage in cyber warfare [**6**, **13**, **98**, **102**], while, some hackers do it for fun.

Microsoft [**27**] proposed a threat model that categorizes threats into six categories.

- Spoofing: An attack the threat actor impersonates a device or user on a network.
- Tampering with data: Data tampering includes unauthorized modifications in the data and adulteration of data in the network.
- Repudiation: Repudiation involves a malicious task performed by the attacker which cannot be validated
- Information Disclosure: The attack which involves revelation of data for unauthorized or unprivileged users
- Denial of service: This type of attack denies service to the legitimate users. This type of attack floods the network with a large number of packets and devours the bandwidth of the target
- Elevation of privilege: In this type of threat, an unprivileged or unauthorized user

gains access to privileged data.

One of the most infamous attacks is the distributed denial of service attack (DDoS). In this dissertation, I present a novel mechanism to limit the impact of DDoS attacks on various network services.

1.1. Distributed Denial of Service Attacks

A DDoS attack is a malicious attempt to disrupt the services or even a network and make them unavailable by swamping the target with a flood of pointless packets. The threat actors achieve their objective by utilizing multiple compromised systems to initiate the attack. These include billions of internet of things (IoT) devices connected to the internet and other networked computers. The attacker achieves this by infecting the devices with a malware which opens a backdoor entry to the infected device. This malware abets the attacker in controlling the device. This compromised device is commonly known as *"zombie"*. A collection of these zombies is referred to as botnet [**3**,**36**]. There are several ways the threat actor could achieve this.

1.1.1. Taxonomy of DDoS

Mirkovic et al. [**81**] present few common classifications of DDoS attacks which include classification based on the degree of automation, based on propagation mechanism, by spoofing technique, by the impact on the victim or victim type. The limitations of the transport and application layer protocols are exploited by the attackers to perform a flooding attack. The most common type attacks pertaining to these layers are

1.1.1.1. Layer 3 (L3) - Layer 4 (L4) Attacks

These attacks utilize the transport layer protocols such as User Datagram Protocol (UDP), Transmission control protocol (TCP) and Internet control message protocol (ICMP) to flood the network.

1.1.1.1.1 Volumetric DDoS attack (Network flooding attack)

Attacks that use an enormous amount of traffic to saturate the bandwidth of the network are referred to as Volumetric DDoS attack. These attacks congest the network, resulting in

a massive amount of packet loss. The service availability of the network service drops, as a result, the end to end throughput of the network drops dramatically. Some of the examples of these type of attacks include UDP flooding and ICMP flooding. In UDP flooding the attacker sends an enormous number of UDP packets to random ports of the victim. The victim's host responds to the packet with an ICMP packet. If a million hosts are targeted, both the UDP and ICMP packets consume the bandwidth of the network. The packet size could be up to 65,536 Bytes.

1.1.1.1.2 Protocol exploitation attack

These type of attacks exploit the vulnerabilities or introduce bugs in the l3 protocols to consume the resources of the victim's device. The TCP's "3-way-handshake" design is vulnerable to such type of attacks. In a TCP 3-way handshake to establish a connection, the client sends a SYN packet, which is acknowledged by the server with a SYN-ACK. The server then waits for the ACK from the client. The attacker misuses this design for their benefit and keeps the server on wait state by sending only SYN packets. As a result, the resources of the server are utilized, dropping the service availability to the legitimate user.

1.1.1.1.3 Reflection/Amplification-based attack

This type of attack is carried out by abusing the functionality of reflectors. Reflectors are host devices that respond to requests. The attackers abuse reflectors by sending a large number of requests with "spoofed IP addresses." The reflectors respond to the requests and send the response to the victims bearing the spoofed IP addresses. The requests sent by the attacker are tiny such as DNS requests. However, the reflectors reply with a larger response (DNS response), consuming the bandwidth of the network. E.g., DNS amplification attack, smurf attack.

1.1.1.2. Application layer or L7 flooding attacks

Attacks based on the exploitation of the layer seven protocols such as HTTP, DNS, and Session initiation protocol (SIP). These attacks consume the resources of the victims such as CPU, memory, and disk bandwidth. These are the most sophisticated type of attack.

Detecting these attacks is challenging, as it is difficult to distinguish between legitimate traffic and attack traffic. Some of the most common types of application flooding attacks are:

1.1.1.2.1 HTTP flood

HTTP flooding attack is a volumetric DDoS attack that consumes the resources of the server by sending an enormous number of HTTP-GET requests to the target server. The server utilizes all of its resources to respond to these requests. Eventually, the target server is saturated and can no longer respond or responds slowly to legitimate users with a large delay.

1.1.1.2.2 Slow HTTP attack:

This type of DDoS attack has the potential to bring down a server with limited resources. This attack is performed by exploiting the stateful operation of the HTTP protocol. The attackers send a large number of HTTP requests to the server slowly. Due to the requirement of the HTTP protocol, the server does not close the socket until it receives the complete request. This slow rate keeps the server's resources waiting and will eventually saturate the server, dropping the service availability. Some of the examples of slow HTTP attack used before including slow headers, slow body (R-U-Dead-Yet) and slow read [**29**, **88**].

1.1.2. Impact of DDoS

DDoS is a malicious attempt which brings networks and various web-services to knees. The services are made unavailable to the users. These attacks not only limit the users to access the services but also potentially damage the reputation of many businesses and companies. According to Verisign [**37**] the impact of DDoS on Business is classified into four categories:

- Revenue loss
- Productivity loss
- Theft
- Reputation damage

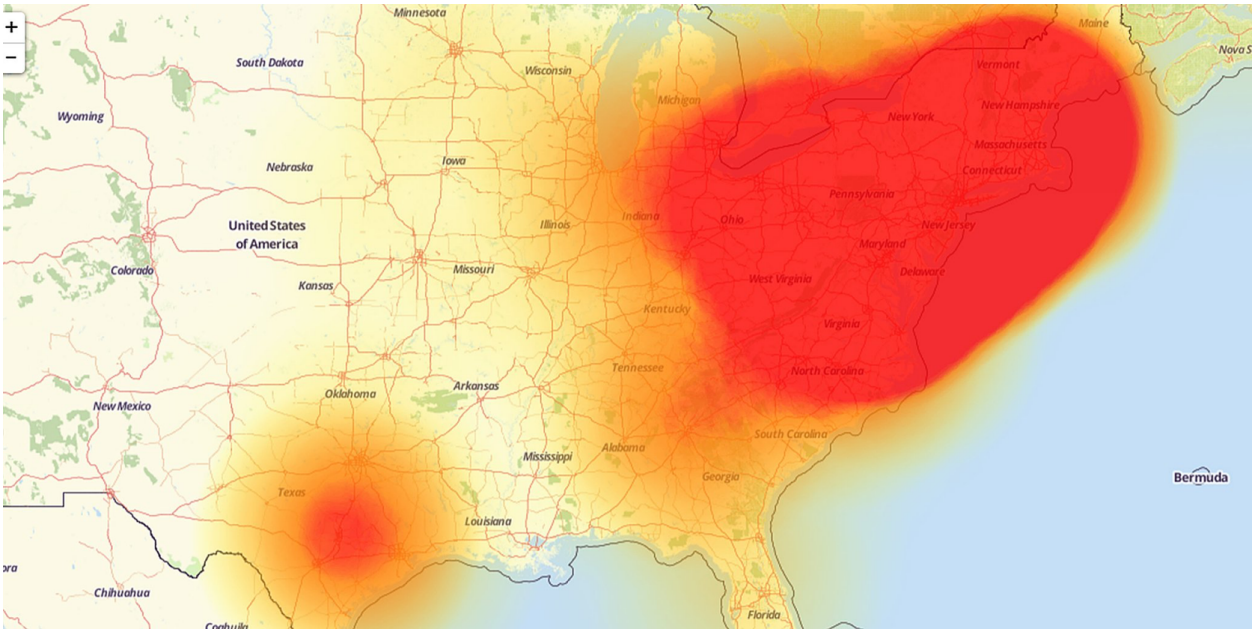

FIGURE 1.1. Internet outage during the Dyn DDoS attack. This data is provided by The Daily Dot. Src:[**7**]

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. The attack rate measured was 600 Gbps. Figure 1.1 shows the impact of the attack on Dyn Inc. During the two hour attack, there was a network outage in a significant portion of the U.S. north-east coast and central Texas. Due to the attack, users were not able to stream music and post their tweets.

Figure 1.2 shows the fail ratio of the Dyn and other DNS service providers. According to [**22**], during the attack the fail rate of the Dyn service increased to 80% and averaged nearly 50%. While the fail ratio of the rest of the services such as Google, Azure, and Cloudflare was as low as 5% - 10%. Figure 1.3 compares the response time of the Dyn service with others. The response time during the attack peaked at nearly 3.5 seconds, which is unacceptable.

1.1.3. Mitigation Techniques

Existing DDoS mitigation mechanisms usually rely on detection techniques. Some of the most common mitigation techniques employed by the internet service providers are as

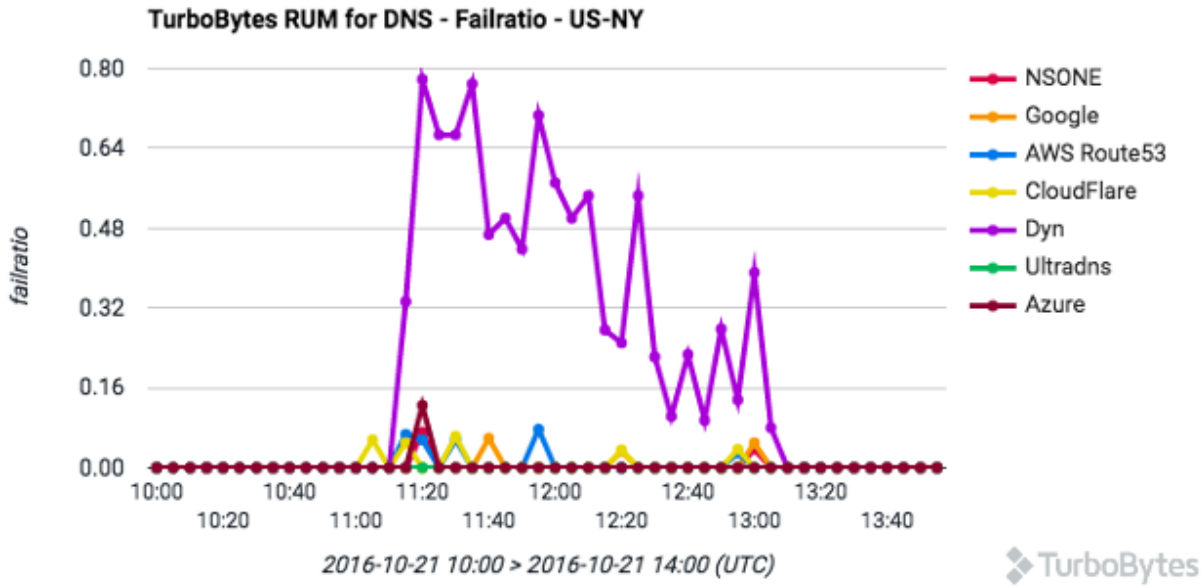

FIGURE 1.2. DNS - fail ratio. This data is provided by TurboBytes. Src:[**22**]

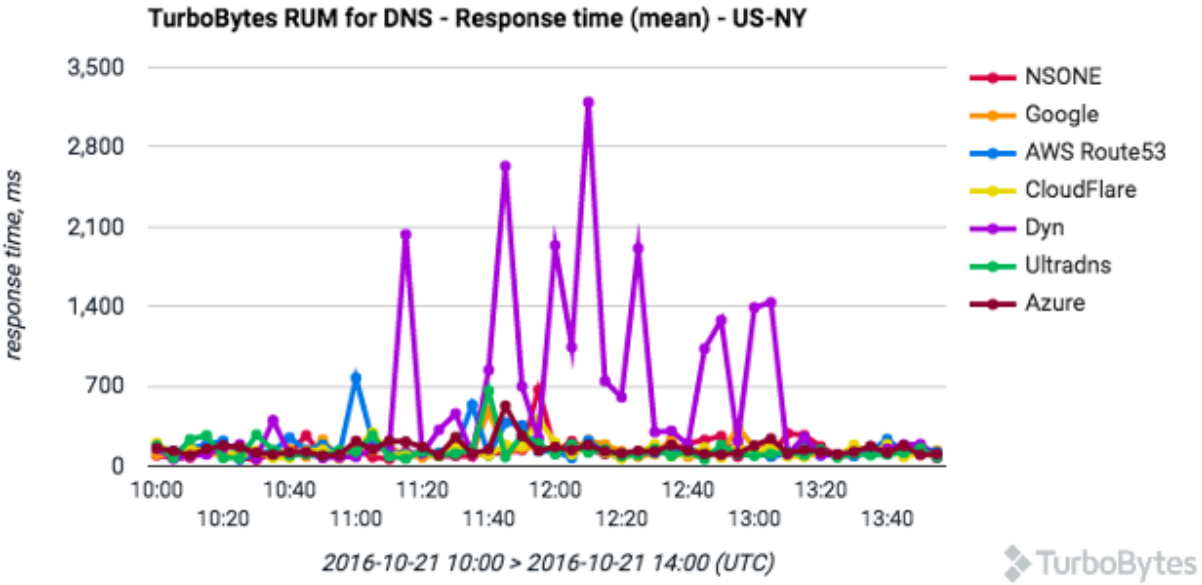

FIGURE 1.3. DNS - mean response time in milliseconds. This data is provided by TurboBytes. Src:[**22**]

follows [**91**].

- *Access Control List:* Acess control lists (ACL) are a set of rules imposed on traffic or data files to control permissions. ACL's are applied in network devices such as routers and firewalls. When an ACL is imposed on a router, the packets arriving at the router are verified against the ACL table. The restrictions can be made based on the source and destination IP address, IP subnet, port numbers and various other tuples.
- *Rate Limiting:* Rate-limiting mechanisms are employed by network systems to control the traffic flow rate. This mechanism limits the number of packets originating from a malicious source identified by various detection algorithms and tools. This control is not only imposed on traffic originating from malicious sources but also aim at specific network service. This limiting technique can be applied to both incoming and outgoing traffic. Restrictions on the traffic are applied based on two parameters "conform action" and "exceed action." The main drawback of this mechanism is that it does not characterize the attack; as a result, some of the malicious traffic find its way to the victim.
- *Combination of Rate-limiting and ACL:* A combination of rate-limiting and ACL also can be employed on the network, which provides the benefit of limiting the malicious traffic while allowing the legitimate traffic. The features of ACL and rate-limiting can be enforced on the traffic.

Saman Taghavi Zargar et al., [**115**] present some of the defense mechanisms against DDoS flooding attacks. Some of the other mitigation techniques include anomaly-based detection [**50**,**65**,**99**] and signature-based detection [**44**,**84**,**93**]. Alan Saied et al. [**94**] present an artificial neural network (ANN) algorithm to detect DDoS attacks. Their algorithm differentiates malicious traffic from legitimate traffic using patterns. Some mechanisms also include detection and filtering of adulterated routers [**66**] based on trust including tor routers [**117**].

Some of the defense mechanisms incorporated before an attack occurs are load bal-

ancing [**67**], server-side configurations such as allowing recursive queries from trusted sources [**112**] reduces DNS amplification attacks and by employing filters such as ingress and egress filter [**97**], packet filters, hop-count, and history based filtering as well.

1.1.4. Motivation

On Feb. 7th, 2000, the first-ever DDoS was recorded [**4**] which was targeted on Amazon, eBay Dell, and CNN. The estimated financial damage was nearly 1.2 billion dollars. DDoS attacks have evolved since then. Almost 20 years later, in 2016 and 2017 attacks the likes of which never seen before were registered. These attacks continue to pose a severe threat to various businesses and consumers. With the growth in the number of devices connected to the internet, these attacks continue to grow in number. Despite a number of security tools, the attacks continue to happen and are bringing various businesses to their knees.

1.1.4.1. Anticipating Attack

Existing static techniques [**91**] focus on detection and mitigation which is a passive approach. These are inadequate to respond to the ever-changing threat landscape. Additionally, detecting malicious traffic from the legitimate traffic is difficult especially when the attacker impersonates a device or a user. Current mitigation techniques and tools may take anywhere from a few hours to a few days to counter the attacks, which is entirely unacceptable.

Dynamic Systems

Networks are complex, dynamic and volatile. With ever-changing network configuration, predicting the route of the packet is very difficult and challenging. Slight changes in the network configurations cause severe disruptions in the network traffic. Understanding the dynamic nature of a network is critical to being able to create and maintain an accurate model.

1.2. Objective

The primary goal of this dissertation is to be able to provide uninterrupted network services in the face of an attack or any failures. It is my aim to capture the dynamics of the network to be able to build an accurate model. An attack disrupts the real-time services in the network such as video and voice, degrading the quality of experience.

My work is interdisciplinary to computer science and control system theory with a focus on system identification and closed-loop feedback control. This methodology can be used to model the dynamics of the network.

1.3. Contributions

In this section, I will highlight the significance and impact of each project in the field of network security. The significant contribution of this dissertation is to applying control theory to achieve service level objectives in the face of an attack.

1.3.1. Building Resilient Systems

DDoS attacks continue to be problematic for websites and service providers and thanks to the Internet of Things (IoT) supporting a rapidly growing number of network-connected devices from refrigerators to thermostats, and so on, this trend is expected to grow in size and complexity. With attacks growing in number, the design of a resilient network is critical. As mentioned by Tammy Butow [**5**], the need for building resilient systems is rising. Resilience is the ability of a system to withstand when provoked by an external disruption. Disruption or disturbance is an abnormal activity that hinders the normalcy of the system. Resilient networks have the ability to provide the desired level of service, despite challenges such as malicious attacks and misconfigurations.

A desirable attribute of resilient networks is the ability to configure the network dynamically. In this manner, feedback control mechanisms play a crucial role. These mechanisms can be used in mitigating the attacks in real time [**52**]. Feedback control is about regulating the system characteristics with disturbance rejection.

My dissertation focuses on designing a robust feedback control mechanism that is both scalable and effective in real-time. These feedback mechanisms offer uninterrupted network service even in the face of an attack. Enforcing these mechanisms also stabilizes the unstable network in real-time. In my approach, I look to system identification to design and validate a model for the complex and dynamic network.

1.3.2. Graceful Degradation

DDoS attack causes severe degradation of network services. Attackers flood the network with pointless packets and congest the network, resulting in a huge amount of packet loss. Mission-critical real-time applications such as surveillance traffic, 911 emergency calls and air traffic control cannot tolerate any packet loss.

My research focuses on employing dynamic and predictive control approaches to reduce the impact of an attack on real-time services. The closed-loop control feedback control mechanisms tackle this issue by degrading the real-time service gracefully to an acceptable level and the stabilize the network in real-time.

1.3.3. Self Configurability

Networks are configured by the internet service providers to provide better QoE to the users. Service level agreements (SLA) are enforced on a network to ensure user satisfaction. Service level agreement is a commitment guaranteed by the internet service providers to the user. These agreements include one or more service level objectives (SLOs), which is a criterion used to evaluate the performance of the service. Enforcing these objectives requires the service providers to meet their requirements.

Existing static configuration settings does not guarantee efficient services with ever-changing network conditions. Employing a feedback mechanism provides the ability to automatically configure the settings such that the QoS metrics of the network is consistent with those specified in the service level agreements.

1.4. Dissertation Roadmap

The roadmap of this dissertation is presented in Figure 1.4.

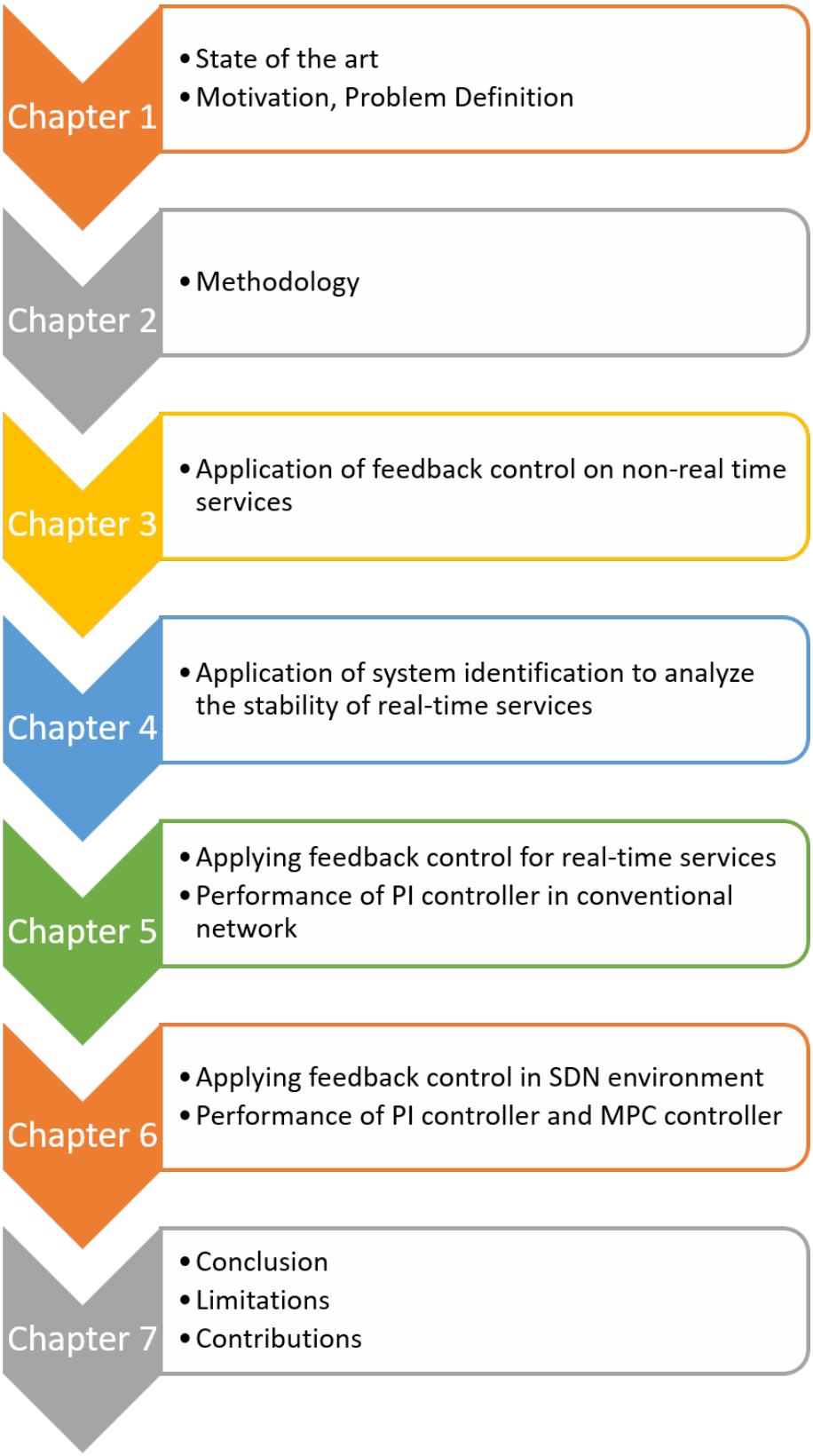

FIGURE 1.4. My dissertation roadmap

CHAPTER 2

METHODOLOGY

2.1. Introduction

Feedback control is a process of controlling or regulating the characteristics of a system. The objective of feedback control is to use the measurements of the system's outputs, such as response times, QoS metrics, jitter, CPU utilization, and throughputs, to achieve desired system states. This objective is achieved by regulating the system control input, such as queue sizes, maximum clients, number of keep-alive and rates. As the measured outputs determine the control inputs of a system and the control inputs drive the output of a system, this phenomenon is called feedback or a closed loop.

Feedback is the data from the output or a portion of the output collected by a sensor and fed to the controller which controls the dynamics of the system. The measured output data is the characteristic of the system to be regulated to the desired value. The control input, which can be dynamically adjusted, influences the output data. The output is also affected by an unmeasured data known as disturbance input, such as a large number of requests.

Feedback control systems are applied in diverse fields including economy, chemical and biological process, biomedical research, and psychology [**46**,**47**,**59**,**64**,**76**,**104**,**106**,**107**,**110**]. They are also employed in our day-to-day lives such as a vehicle cruise control system which achieves the desired speed by controlling the accelerator pedal using the measurement fed by the speedometer. In this scenario, the control input is the accelerator pedal adjustments, and the measured output is the speed. The desired speed is maintained even when the vehicle is affected by external disturbances, such as friction from the road, slope of hills, and other terrains. Another example is a thermostat, used to control the temperature of a room by controlling the speed of the fan. When the room temperature is less than the desired temperature, the controller increases the fan speed.

These feedback concepts can be applied to computing systems as well [**62**, **63**, **78**, **90**,

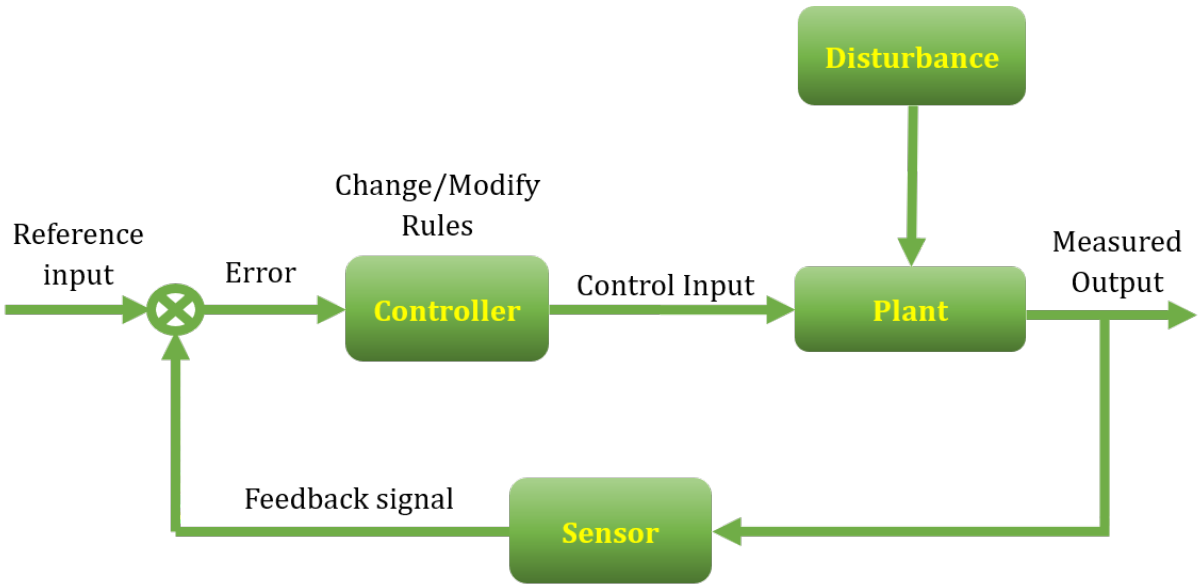

FIGURE 2.1. Block diagram of a feedback control system. The reference input is the desired value of the measured system output. The sensor feeds back the measured current output to the controller. The controller detects the difference between the reference input and the feedback signal and drives the system with an adjusted control input so that the measured output matches the reference input. The disturbance input disrupts the system, unstabilizing the stable system.

105]. For example, to provide a better quality of experience, the CPU utilization of servers should be at no greater 70% [**33**]. The CPU utilization is directly proportional to the number of processes running in the server, generated by the number of requests or workload. So, to maintain the server utilization rate, the workload should be controlled. Hence, the control input is the maximum number of connections permitted by the server. The control input can be regulated by a controller to affect the utilization.

The block diagram of a closed-loop feedback control system is shown in Figure 2.1. The controller which is the brain of the entire system detects the error from the feedback and regulates the system output to the desired state. The essential components of a closed-loop feedback control system shown in Figure 2.1 are:

- *Plant or System:* The system such as the network or the server which is to be

controlled

- *Control input:* The input to the system that affects the behavior of the system.
- *Measured ouput:* The quantifiable data of the system using which the performance of the system is defined.
- *Reference input:* The desired value of the measured output such as CPU and Memory Utilization, and other QoS metrics.
- *Error:* It's the difference between the reference input
- *Controller:* The essential component of the feedback control system which provides an input to the system to adjust the output to the reference input. The controller detects the irregularities in the measured output based on the error signal.
- *Disturbance input:* The unmeasured input that affects the measured output correspondingly as the control input

The principal feature of feedback control is that we can achieve the desired value automatically by providing the reference input instead of adjusting the input to the system, which is time-consuming and requires significant skill and knowledge. The other main feature of feedback control is disturbance rejection, i.e., The desired value. The disturbance input is like a second input to the system that disrupts the system's normal working conditions. An example of disturbance input in computing systems is an attack scenario such as a worm attack [**52**] and a DDoS attack where the number of requests increases sharply.

2.2. System Identification

To build a robust feedback control mechanism, identifying the dynamics of a system is critical and required. System identification is the process of constructing mathematical models of dynamic systems using the statistical methods [**80**]. In this process, models are developed using measurements from inputs and outputs of a system. These inputs and outputs determine the behavior of the system. Inputs are the signals which can manipulate the characteristics of a system. Outputs are the signals that are observed. An external stimulus, known as a disturbance, also affects the system. These signals can be directly measured or observed through their impact on the measured output.

To determine all the internal properties or features of a system is challenging. A model generated by the system identification technique tries to emulate critical features of the system behavior. System identification techniques are classified into two different ways:

- prior modeling or white-box identification technique: In this approach the parameters of the model are estimated from the data.
- black-box identification technique: This is a data-driven approach. The system dynamics are identified only by using experimental data without having any prior knowledge. This technique is the most commonly used. This dissertation uses black-box method to identify the system characteristics.
- grey-box identification technique: This technique can be used when certain features of the system are known but not entirely. This technique is based on both prior knowledge and data collected.

System identification technique estimates the parameters of a black or grey model based on the observed input-output data. The applicability of system identification techniques did expand in diverse fields including computing systems. System identification technique serves various purposes including:

- To design robust feedback control systems and control strategies
- To analyze the dynamics of the system
- To forecast the functioning of the system

The following steps are involved in constructing a model:

(1) *Collect data.* The inputs and outputs of the system are recorded from several experiments where we determine which signals to measure. The selection of inputs and outputs is very critical, as they define the dynamics of the system.

(2) *Select a model.* A set of models are selected which look suitable to define the system characteristic. This step is the most important and the most challenging. Models can be constructed with a priori knowledge and engineering intuition and insight. Models can also be constructed with basic and other known relations between the input and output data. In most of the cases, where prior knowledge is unavailable, models are identified as a black-box adjusting the fit to the data.

(3) *Identify the model.* The system is identified at this step. The best model is determined from the set of candidate models. After determining the best model, the parameters of the model are identified and recorded.

(4) *Model Validation.* After identifying the model and arriving at a particular model, we validate the model with a fit criterion. We test whether the model is good enough to determine the system dynamics with a newly observed data (different from the data used for modeling).

2.2.1. System Identification Loop

The flow diagram of the system identification procedure is shown in Figure 2.2. First, collect data from a selected set of experiments. Then, select a model from the candidate model set, then picking the best model from the set. The model obtained must be validated using a fit criterion. If the selected model fails, go back and revise the steps from step 2.

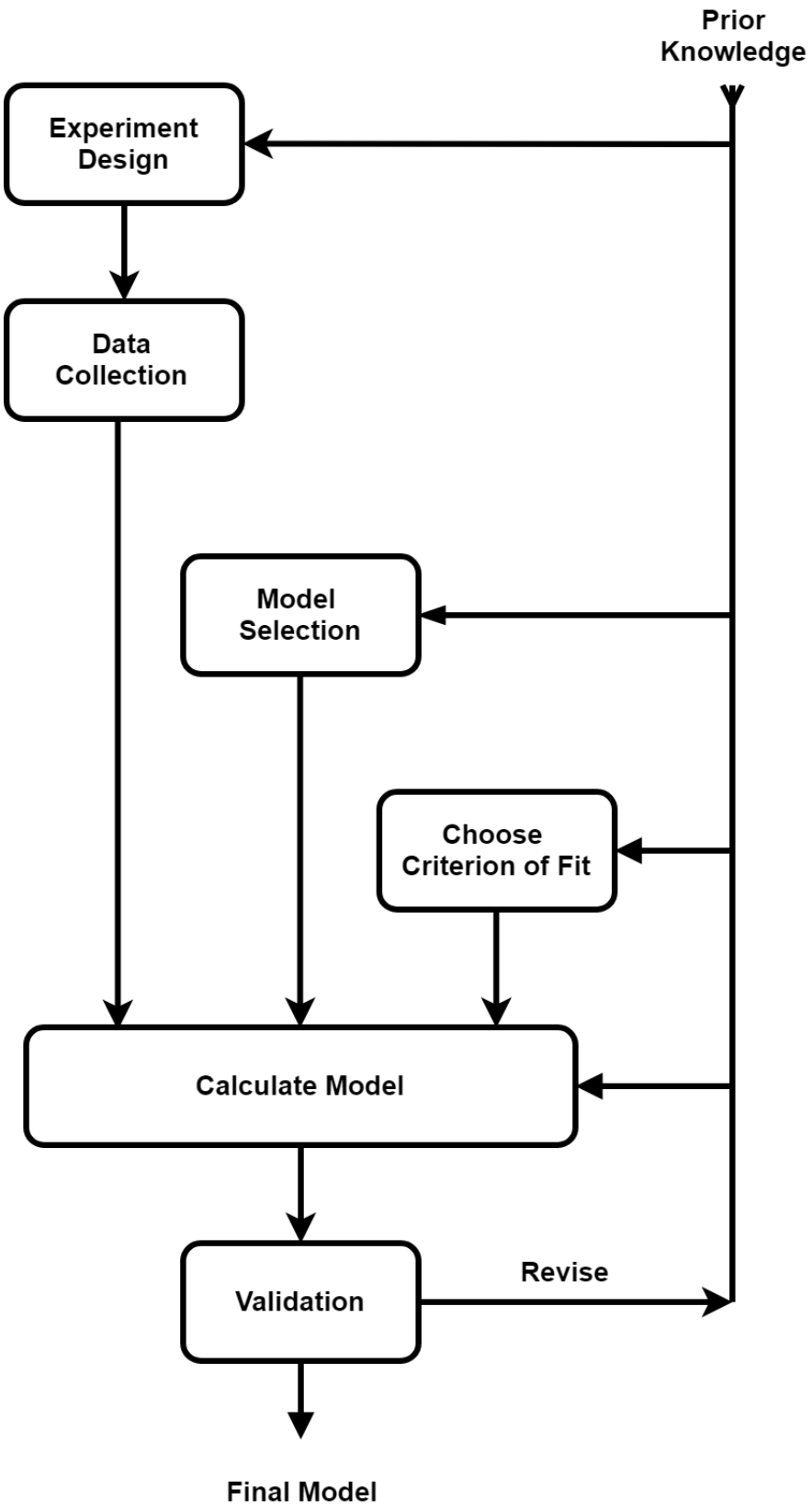

FIGURE 2.2. System identification loop [**80**]

2.3. Control Analysis and Design

The main objectives focused on designing a controller are:

- **Regulatory control.** To ensure that the measured output matches the reference input. For example, the Quality of Service metrics such as jitter and packet loss should be maintained at less than 10ms and 5% respectively.
- **Disturbance rejection.** To ensure that the disturbances introduced on the system should not significantly impact the measured output. For example, when a stable network is attacked with a large number of packets the network congests, dropping the QoS of the traffic. However, a good controller ensures the QoS of the real-time traffic is maintained in spite of an attack.
- **Optimization.** To obtain the optimum value of the measured output.

2.3.1. Control Loops

Fundamentally two types of control loops exist:

- **Open loop control**: The control system where the controller regulates the system or the process independent of the system output. E.g., to constantly heat a room for a certain period, regardless of the temperature of the room.
- **Closed loop control**: The control system where the controller regulates the system by utilizing the feedback from the system output. E.g., In the similar analogy mentioned for open-loop, if a thermostat is used, the sensor detects the room temperature. This feedback is used by the controller to regulate the temperature to reach the desired value.

However, this dissertation will be focused on closed-loop control.

2.3.2. Proportional Integral Derivative Control

A proportional-integral-derivative controller (PID controller) is a closed-loop control mechanism often used in many industrial chemical and computing systems [**40**, **43**, **61**, **75**, **101**,**109**,**116**]. In a closed loop control system the current output measured from the system, also known as a process variable, is fed back to the controller along with the desired reference

value. The reference value is also called as a setpoint. The controller continuously calculates the error e(t) measured from the process variable and the setpoint and applies a correction to the system based on the P, I and D terms. The error is fed to all the variables individually. These three variables determine the controller's behavior. The PID controller architecture [**19**] is shown in Figure 2.3. The general equation of a PID controller is given below:

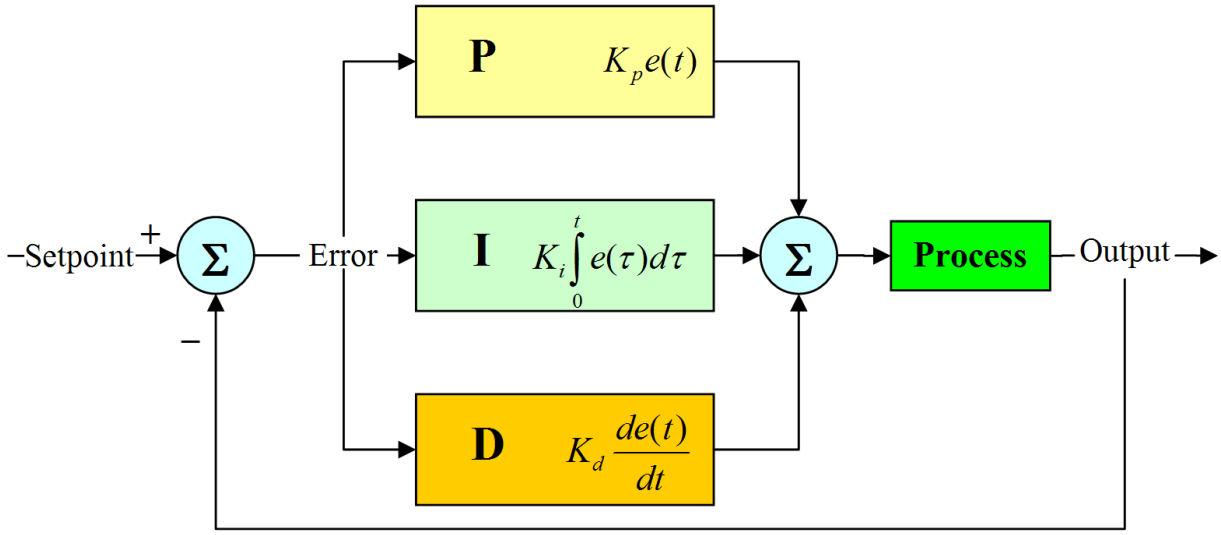

FIGURE 2.3. Proportional, integral and derivitive controller architecture [**19**]. The process is the plant or the system under control. The setpoint is the desired value of the system output. The error is the difference calculated from the current measured system. Src: [**19**]

$$u(t) = K_p e(t) + K_i \int_0^{t'} e(t')dt' + K_d \frac{de(t)}{dt} \tag{1}$$

where,

u(t)— control input at time t

e(t)— error, which is the difference between measured output and setpoint at time t.

K_p : proportional gain.

K_i : integral gain.

K_d : differential gain.

2.3.2.1. Components of PID Control System

The proportional (K_p) variable adjusts the system output proportionally to the error signal by controlling the proportional gain of the controller. Isolated use of a proportional controller might help in reducing the error, but the final output might result in oscillations such as an on-off signal. These oscillations can be reduced by the integral variable (K_i) by rectifying the accumulation of the error. The integral The derivative term (K_d) estimates the future trend of the error based on the current rate of change of the output. The derivative tries to neutralize the overshoot caused by the P and I terms.

2.3.2.2. Tuning

Tuning a PID controller is to adjust the control parameter, i.e., the proportional gain (K_p), integral gain (K_i) and the differential gain (K_d), such that the output closely follows the setpoint or the reference value. Due to the on-off characteristic of the P term, higher K_p value results in oscillation. As mentioned earlier, the oscillations are controlled by the integral term; hence, higher K_i value implies that the output converges to setpoint quickly. If this reaction very fast, increasing the K_d value will slow down the response and ensures the system is under control.

2.3.3. Model Predictive Control

Model Predictive Controller (MPC) is an exceptional feedback control system that uses a model to predict the future outputs of a process. The main components of MPC are presented in this section. A good understanding of the use of these components is the key to effective implementation of the various MPC algorithms.

2.3.3.1. Modeling and Prediction

This component is the core component of the prediction control mechanism. In order to automate the predictions of the future behavior of a system, a model that captures the dynamics of the system is required. The model should be able to show the correlation of the output on the current control input or the measured variable and the future inputs.

Some of the important questions that arise while performing predictions are :

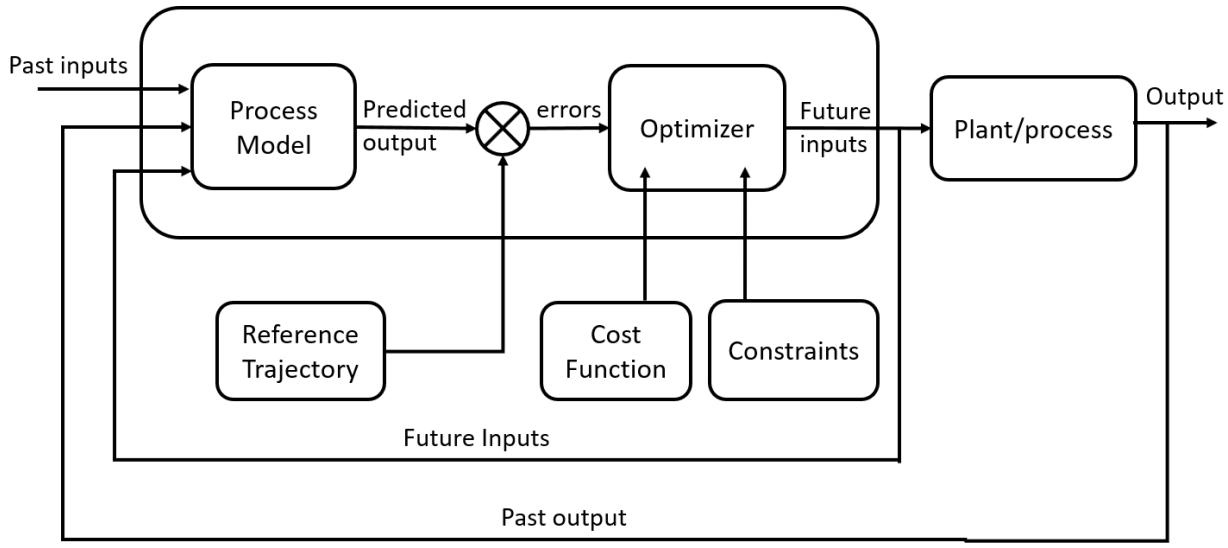

FIGURE 2.4. Proportional, integral and derivitive controller architecture [**19**]. The process is the plant or the system under control. The setpoint is the desired value of the system output. The error is the difference calculated from the current measured system

(1) What is the significance of prediction?

(2) How far should we predict in the future?

(3) How can we make accurate predictions?

The important requirements to consider while modeling is:

(1) The model should be easy to form predictions, ideally linear

(2) The model parameters should be easy to identify

(3) The model should give accurate predictions.

2.3.3.2. Receding Horizon

To understand the receding horizon, let's consider an analogy of driving. While we are driving, we can observe the next few yards of the road ahead and anticipate any roadblocks or pit holes. As we keep driving, the next few yards of the road keeps moving away at the same speed we are approaching it, which is receding horizon, that is the information of the next few yards of the road conditions are continuously fed to update our driving decisions. MPC works in a similar way; where the behavior of the system is predicted in the future for

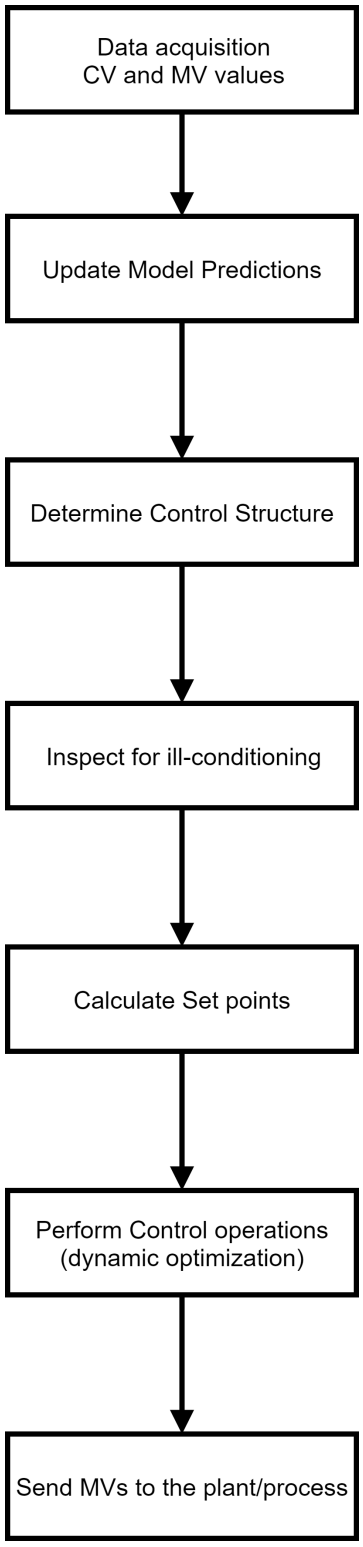

FIGURE 2.5. Flowchart for MPC calculations [**96**]

a certain number of steps. The input trajectory is updated as the controller is fed with new information. The horizon considered should contain the essential dynamics of the system.

2.3.3.3. Tuning

Much of MPC tuning involves in the selection of the appropriate cost function since the controller solves the online optimization problem of a cost function. Tuning involves adjusting the objective function constraints. Following are some of the tuning parameters present in the MPC controller provided by apmonitor [**12**]:

Application tuning:

- IMODE = 6 or 9 for model predictive control
- CV_TYPE = CV type with 1=$l_1 - norm$, 2=squared error
- DIAGLEVEL = diagnostic level (0-10) for solution information
- MAX_ITER = maximum iterations
- MAX_TIME = maximum time before stopping
- MV_TYPE = Set default MV type with 0=zero-order hold, 1=linear interpolation

Manipulated Variable (MV) tuning:

- COST = (+) minimize MV, (-) maximize MV
- DCOST = penalty for MV movement
- DMAX = maximum that MV can move each cycle
- FSTATUS = feedback status with 1=measured, 0=off
- LOWER = lower MV bound
- MV_TYPE = MV type with 0=zero-order hold, 1=linear interpolation
- STATUS = turn on (1) or off (0) MV
- UPPER = upper MV bound

Controlled Variable (CV) tuning:

- COST = (+) minimize CV, (-) maximize CV
- FSTATUS = feedback status with 1=measured, 0=off
- SP = set point with CV_TYPE = 2

- SPLO = lower set point with CV_TYPE = 1
- SPHI = upper set point with CV_TYPE = 1
- STATUS = turn on (1) or off (0) CV
- TAU = reference trajectory time-constant
- TR_INIT = trajectory type, 0=dead-band, 1,2=trajectory
- TR_OPEN = opening at initial point of trajectory compared to end

2.3.3.4. Control Design for Multivariable Systems

The major advantage of the MPC algorithm is it handles multi-variable (or MIMO) systems, unlike PID. The fact that PID uses relatively very little information about the dynamics of the plant, designing a PID mechanism for a highly interactive multivariable system is challenging. Although, if a PID solution is developed for a MIMO system, it is often inaccurate and can be detuned. On the other hand, MPC with an integrated plant captures the dynamics of the plant effectively.

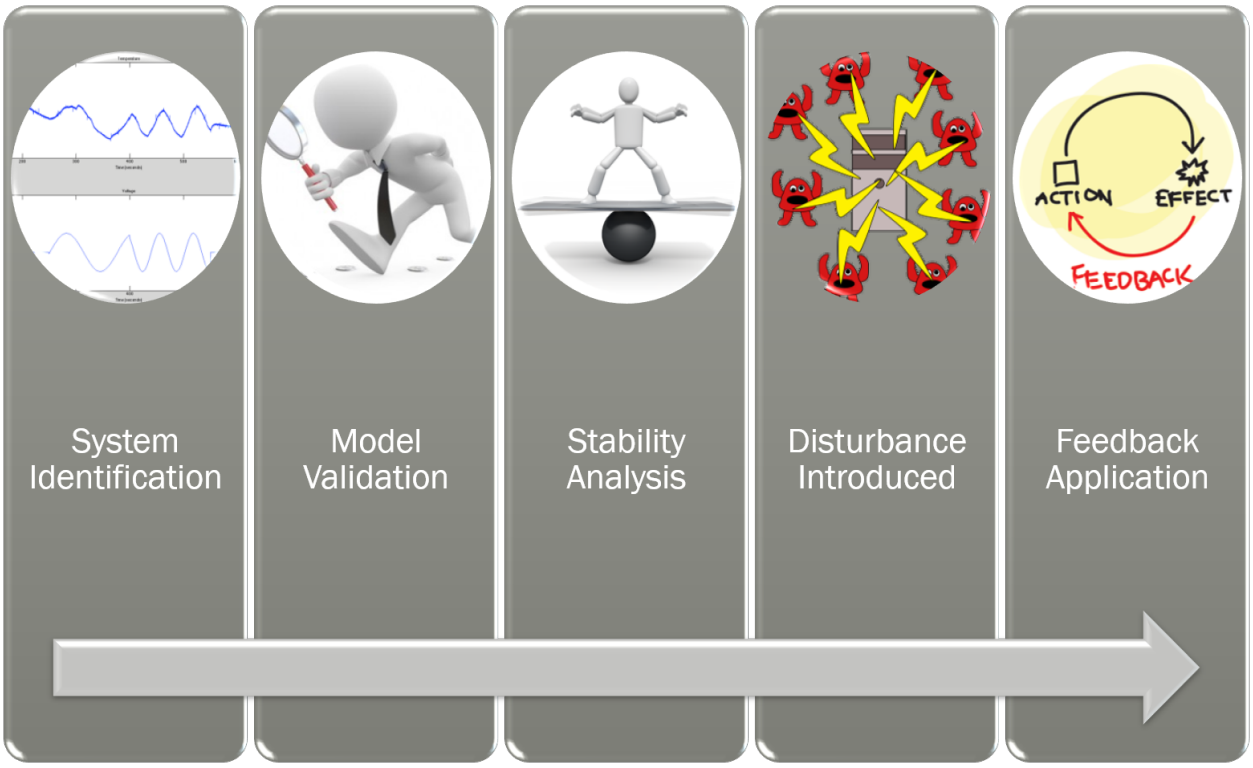

FIGURE 2.6. Steps involved in building the closed-loop feedback control mechanism

2.4. Methodology Implementation Steps

Figure 2.6 shows the steps involved in designing and building a robust feedback mechanism for the network. Initially, the system is identified using the input-output data collected from the network. The data collected varies for different types of services. For a network with non-real time traffic, the input data comprises a number of requests, a number of clients, and the number of hits (or hit-rate). The output data collected are the performance metrics of the service and the system such as response time, CPU utilization of the web server, and memory utilization. Whereas, the input data for real-time services such as voice and video traffic consists of bitrate, packet inter-arrival time, and packet size. The output data is the QoS metrics of the services such as jitter, packet loss, and bitrate. A model is identified from the data collected from the network using system identification toolbox present in the Matlab [**28**].

After identifying the model, it is validated using a goodness of fit test which uses a root

mean squared error. The data used for validation is collected from the same environment, but from a different instance of time. After ascertaining the model, stability analysis of the model is conducted where the model stability is quantified by observing the location of poles and zeros. Poles and zeros are the roots of the denominator and numerator of the transfer function, respectively. Network stability is defined by the QoS metrics measured by the output of the system. If the measured outputs of the system are not within a range of the desired values, the system is tagged as unstable. For instance, in a network if the response time of a web service is more than 2 seconds or if the jitter of the real-time service is greater than 40ms, the network is considered as unstable.

Once the stability of the model is defined, an attack is simulated, to disrupt the network services and capture the characteristics of the network under attack. Finally, the controller is designed to limit the impact of the attack and stabilize the system or the network. Various parameters quantify the performance of the controller: 1) stability analysis, if the measured output is within the desired region, 2) settling time, how fast the network services can be restored, and 2) accuracy if the measured network output converges to the reference values.

CHAPTER 3

SYSTEM IDENTIFICATION OF A MULTI-INPUT MULTI-OUTPUT NETWORK USING HAMMERSTEIN-WIENER MODEL

This chapter presents the dynamic modeling of a traditional network using black-box system identification method. First, we obtain a multi-input-multi-output non-linear Hammerstein-Wiener model with performance metrics such as response time and CPU utilization. Next, we model the output as a piece-wise linear model based on the observations from the experimental data. Last, we validate the model with test data by conducting various experiments such as varying the network topologies and the increasing the rate of requests to the network.

In our approach, we identify a non-linear model for the network. We use a technique called a Hammerstein-Wiener model to estimate the dynamics of the network. The Hammerstein-Wiener model [**111**] consists of a linear block surrounded by two non-linear blocks as shown in Figure 3.1. We use a black-box approach to identify the network. We also observed that the output of the system is piece-wise linear to the inputs.

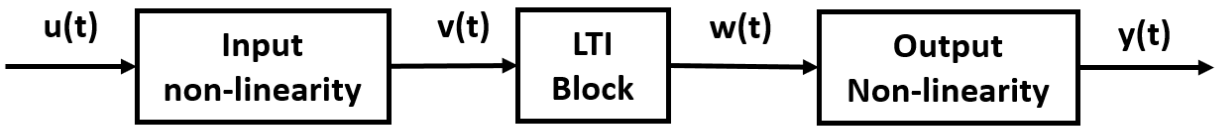

FIGURE 3.1. Block diagram of the Hammerstein Wiener model implemented [**111**]

3.1. System Identification

We considered the complete network, comprising of clients, network devices such as routers, switches, and firewall, moreover, web server as a plant. We model this plant using the system identification toolbox present in Matlab [**28**]. The input parameters to the plant are

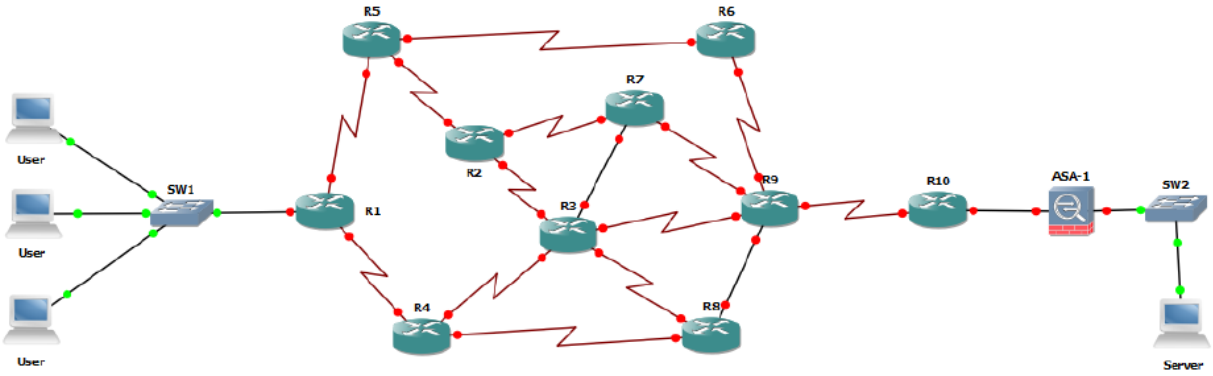

FIGURE 3.2. Experimental network topology: network elements connected in a mesh topology. All the network devices are configured with open shortest path first (OSPF) routing protocol. The routers are configured with a round robin load balancing algorithm to share the load equally among the links.

the request rate and the number of users requesting the web server. The output parameters measured are Quality of Service metrics such as response time and the performance metrics such as CPU and memory utilization of the devices. For this experiment, we are focusing only on the metrics of the web server.

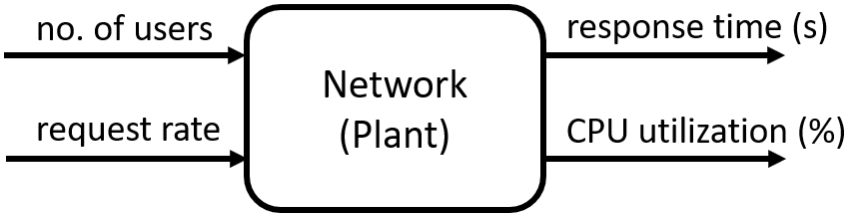

FIGURE 3.3. Open-loop system. The input variables measured are number of users and request rate; The output variables measured are response time to access the web page and cpu utilization of the web server.

3.1.1. Assumptions

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

(1) We consider two inputs; one is the number of requests, and the other is the number of users requesting the web server. The users and the requests increase linearly.

(2) The user arrival rate is defined as the rate of users/ new users per second.

(3) The total number of users is limited to 3,000

(4) The web server serves a single static page of size 200 KB. The maximum requests to the server are limited to 650 requests/sec to ensure that all the requests are served without rejections.

(5) We consider two outputs, response time and CPU utilization

(6) We assume that there is no background traffic, and the firewall allows all the web traffic.

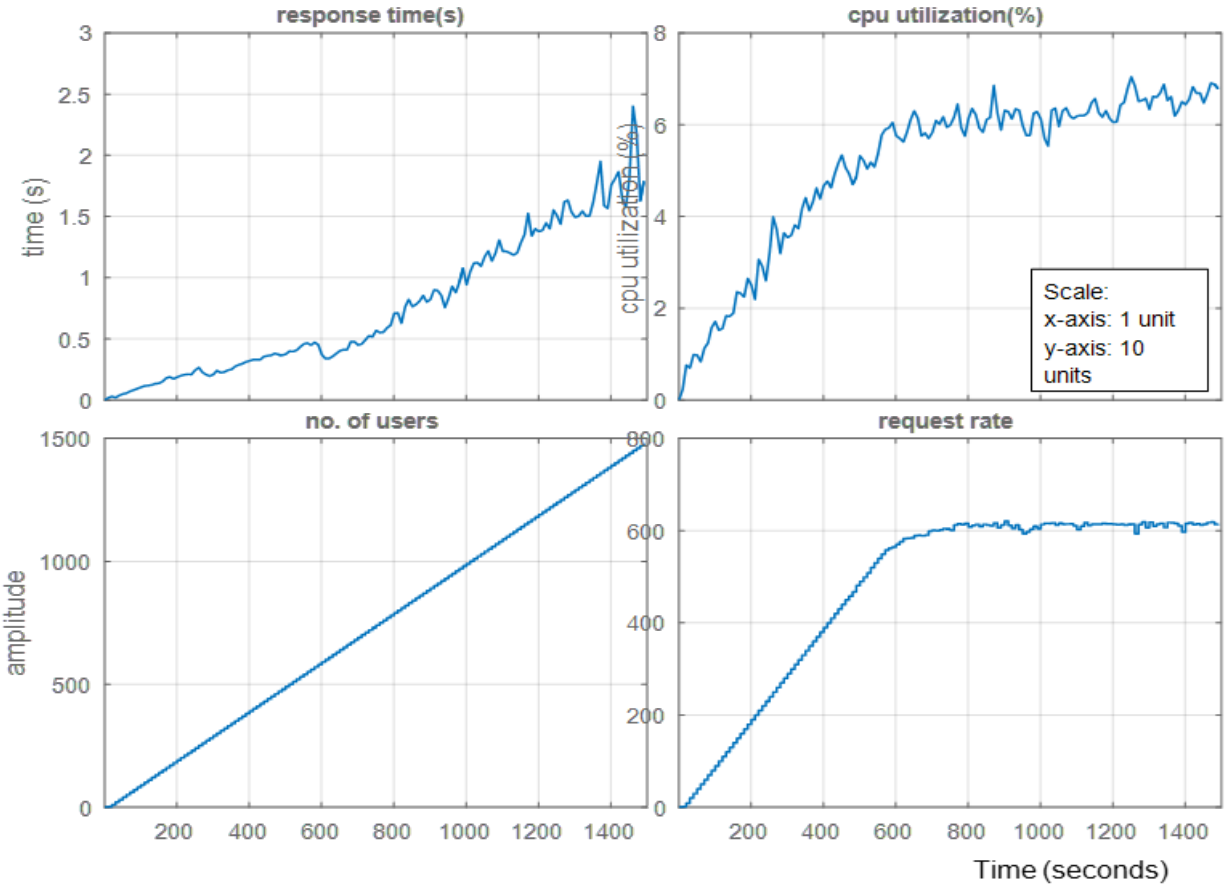

FIGURE 3.4. Input-output data collected from the network as shown in Figure 3.2. The input variables considered are number of users and request rate. The output measured from the plant are response time to access the web page and the cpu utilization of the server. The figure illustrates the non-linearity nature of the outputs.

3.1.2. Linear Regions of the System

The input-output data are collected with a sampling interval of 10 seconds as shown in Figure 3.4. The figure illustrates that the outputs measured are non-linear to the inputs. We observed that in specific regions the response time is linear to the rate of requests as well as to the arrival rate of the users/clients. Hence, we conducted experiments to measure the linear operating regions.

The experiments were carried out by constantly increasing the user rate. The corresponding response time and CPU utilization were measured and plotted. From the Figure 3.4, it is clear that the response time is non-linear to the request rate. However, if we carefully consider specific regions, from 1 second to 15 seconds and also the region from 17 seconds to 30 seconds, we observe that the response time is linear to the user arrival rate. Hence we derived to a conclusion that the response time is piece-wise linear to the request rate.

3.1.3. State Variables

We considered the state variables such as the average response time and CPU utilization measured in seconds and percentage. These variables are measured with a sampling time interval of 1 second, 5 seconds and 10 seconds. From Figure 3.4 we can observe that the response time is piece-wise linear to the request rate; hence we model the system as a piece-wise linear system. From the figure, it is also clear that both the outputs, response time and CPU utilization are dependent on both the inputs at any instant.

3.1.4. Experimental Environment

The experimental study for system identification is conducted in our lab environment. The setup consists of a standalone server, ten physical nodes (Cisco catalyst devices), a firewall and three machines to simulate users/clients. A dedicated machine was used to run the Web server. We used Apache JMeter [6] load tester to generate HTTPS requests to the server. The details of the setup are as follows:

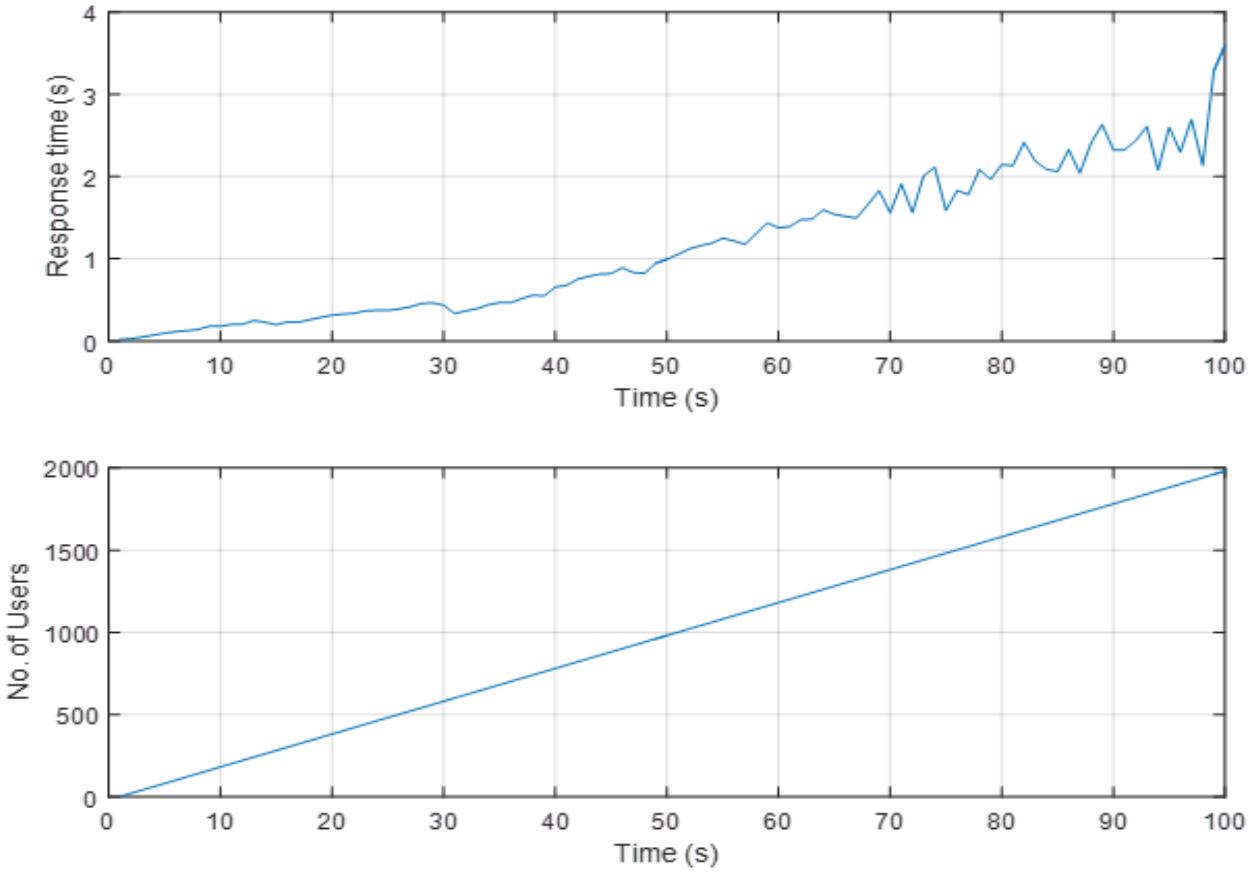

FIGURE 3.5. (a) no of users requesting the web-server and (b) corresponding response time of the network; The system is excited with linearly increasing users and the corresponding response time is collected

3.1.5. Client

Linux containers (LXC) [5] were used to emulate a real-world scenario. LXC is an operating system level virtualization to run multiple Linux hosts on a single system. These containers resemble as clients to send requests to the Web Server. These containers are connected through a bridge interface in the same subnet. The number of users is linearly increased and the corresponding response time to access the web page and the CPU utilization of the server were recorded.

3.1.6. Server

A standalone machine is used to host Apache HTTP web server [7]. The following are the specs of the hosted server: (i) Intel Xenon processor (4 cores); (ii) 32 GB RAM; The Apache server is configured to accept 10,000 requests/sec. The throughput of the server was limited to 650 requests/sec. This configuration is to ensure that all the requests are served without being rejected. The keepalive is enabled in the configuration to reduce the number of connections to ensure not to overload the server.

3.1.7. Generation of Traffic

Apache JMeter load tester is used to generate HTTPS requests and also to measure the performance of the server. Distributed testing [refer] scenario of the Jmeter is used to send multiple HTTPS requests from the Linux containers. The way the distributed testing works is one master node/controller initiates the test on multiple slave systems. The slave systems, i.e., containers, in this case, send the quality of service (QoS) performance metric data such as response time, latency, and a number of request samples to the master controller periodically. The master controller also collects the performance metrics of the server such as CPU utilization and memory utilization. The master controller resembles a sensor which records the output data periodically.

3.2. Results

Two different types of experiments were conducted. In both the experiments the server was loaded with 3,000 users with a fixed arrival rate. The request rate was limited to 650 requests per second so that all the requests are served without any rejections. Following are the details of the experiments:

3.2.1. Model

Considering the non-linear nature of the response time and CPU utilization, we selected a non-linear model. The model that was selected was the Hammerstein-Wiener model. As mentioned in section IV B we observed that the output of response time was piece-wise linear. Consequently, the output nonlinearity of the response time was selected as piece-wise

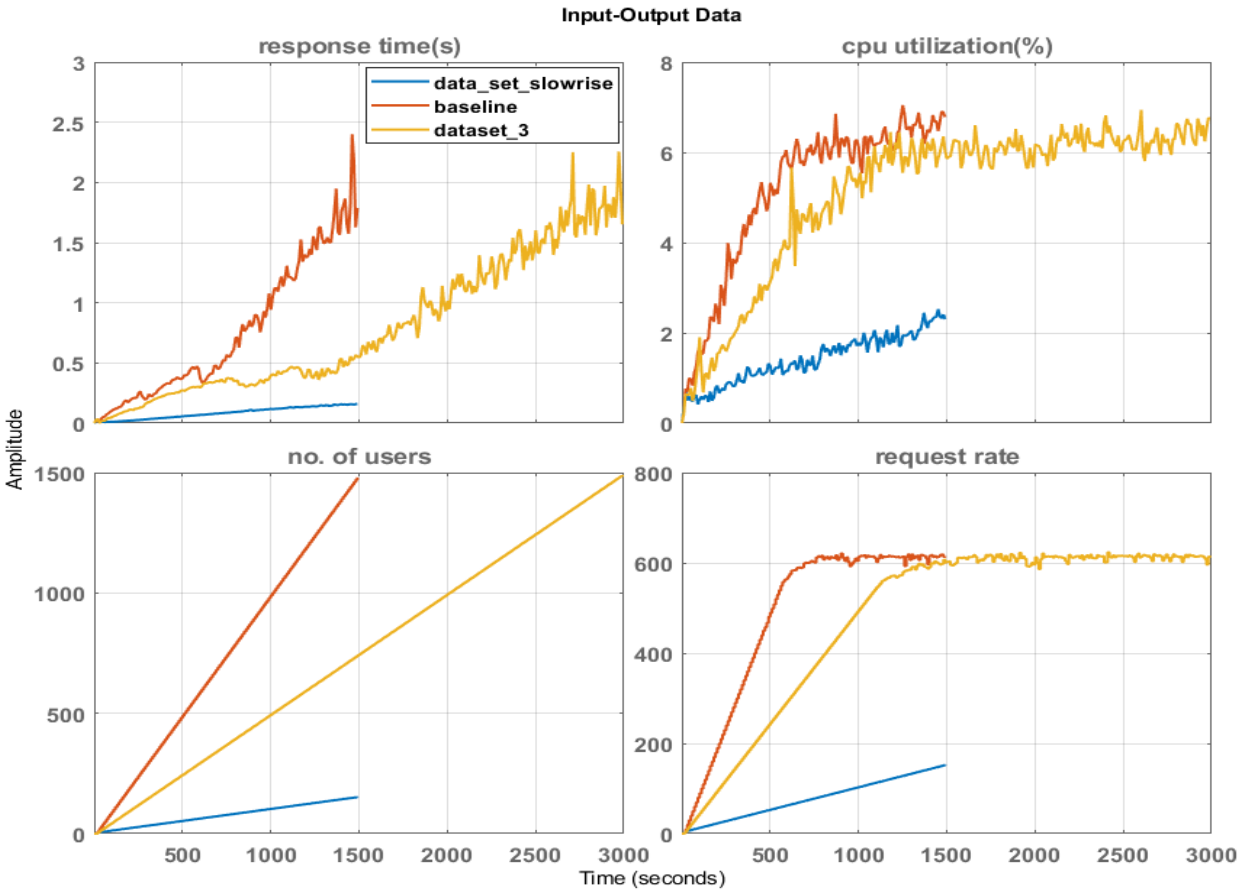

FIGURE 3.6. Input-output data profile. The input to the network is varied by altering the number of users and request rate or the hit rate. The corresponding response time and CPU utilization of the web-server are collected. As the number of users and the request rate increase, the response time of the request and the CPU utilization of the server increase linearly. The request rate saturates at 600 requests; as a result, the CPU of the web-server saturates at 60%

linear with four units. The input to the system was a ramp function, i.e., the arrival rate of users, hence we selected the non-linearity as a one-dimensional polynomial with two units.

3.2.2. Validation

The baseline data is collected from the network shown in Figure 3.2 . The users are limited to a maximum of 3,000 while capturing the data. The data was collected with a

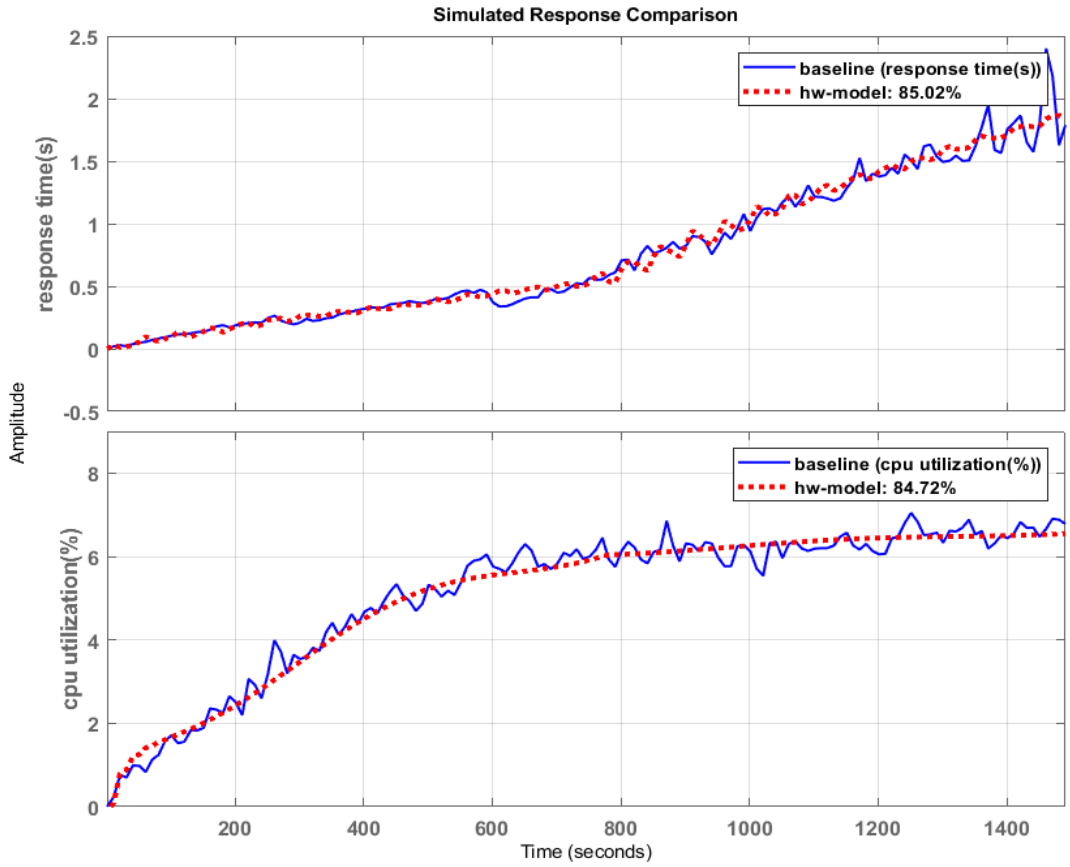

FIGURE 3.7. Baseline data collected from the network Figure 3.2. The system identification model is obtained from the data sample. The model (nlhw27) was verified against the collected data. A fit percent of 85% for the response time and 85% of cpu utilization was obtained. The data was collected at sampling time interval of 10 seconds.

sampling time interval of 1 second, 5 seconds and 10 seconds. This observed data from the baseline is used for system identification to estimate the model.

After obtaining the model, we validate the model with the experimental data collected from the experiments mentioned in section IV. We observe that the predicted model closely follows the response time and CPU utilization collected from the experimental setup.

Normalized root mean square error (NRMSE) was used to validate the goodness of fit value. The NRSME fit value provides the goodness of fit value of predicted model over

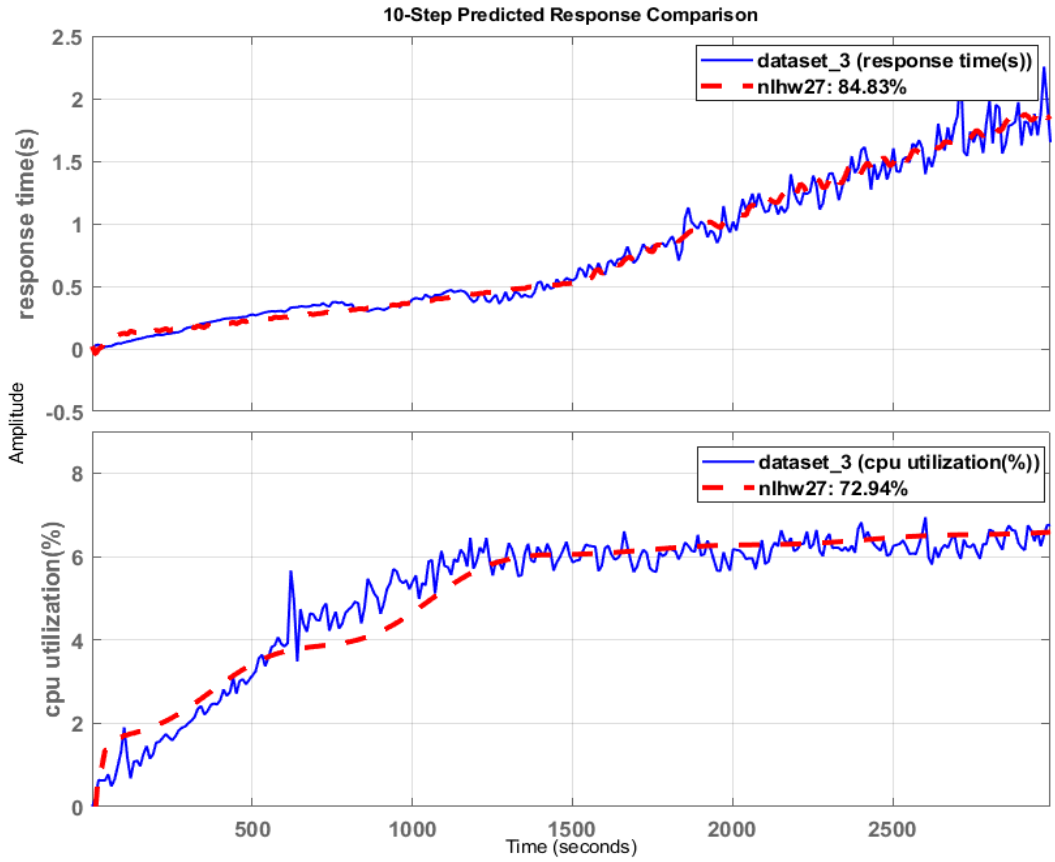

FIGURE 3.8. The fit accuracy of the predicted Hammerstein-Weiner model against the data sample collected with medium arrival rate of the users. The model closely follows the response time and the CPU utilization.

the experimental data. The derived model gives an average fit accuracy of 87% to the experimental data.

Experiment 1: The Figure 3.7 and Figure 3.8 show the fit accuracy of the model against the validation data collected from the network by varying the arrival rate of the users. The model proves to be accurate and consistent irrespective of change in the user arrival rate. However, the CPU utilization prediction was low when a staircase step input was provided to the network. The model did not yield a good accuracy for the step input. The sudden rise in a number of connections per second overloaded the server resulting in an abrupt increase in the CPU utilization of the server. Figure 3.9 shows the model fit accuracy

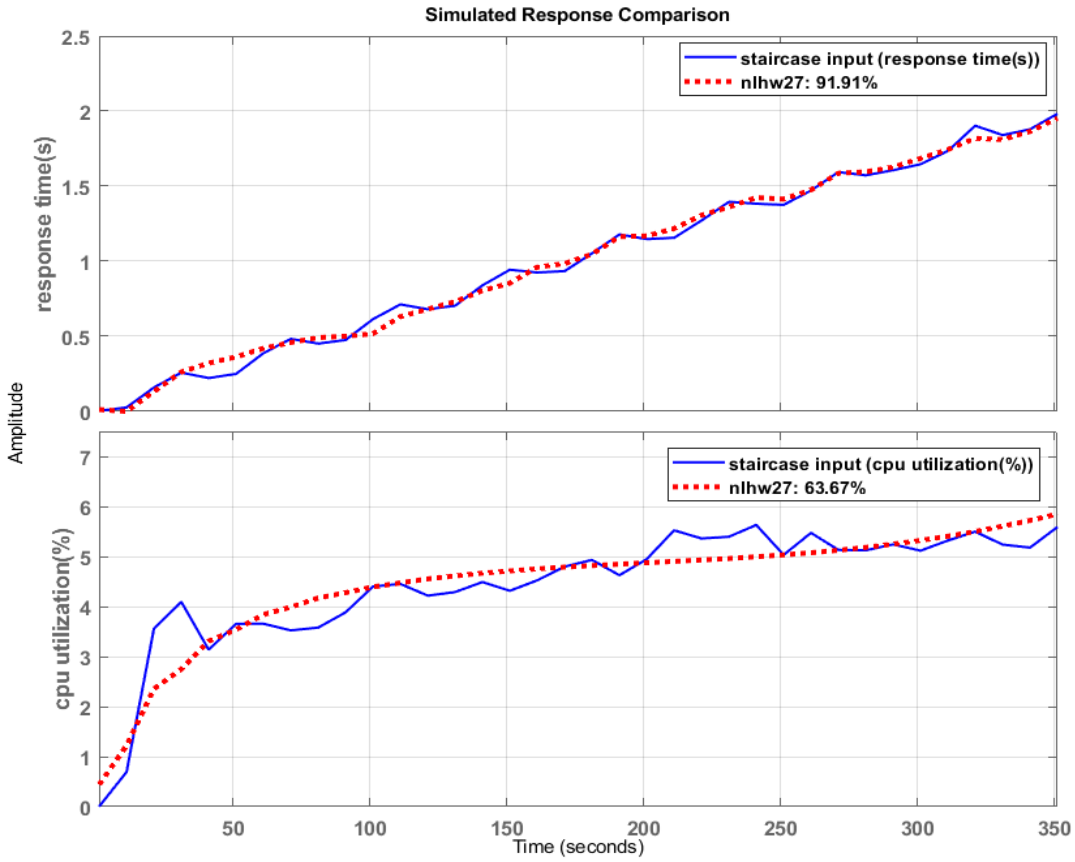

FIGURE 3.9. Staircase step input response; The model closely follows the output, i.e., the response time and the CPU utilization. The drop in the fit of the CPU utilization is because the model is trying to fit all the points; however, it captures the trend of the CPU

with the staircase step input.

Experiment 2: Figure 3.10 shows the fit accuracy of the model for the network. The model fit percent was similar to the baseline fit which proves that regardless of a change in the network topology, the model is suitable for the network.

The following table summarizes the goodness of fit for the predicted Hammerstein-Wiener model against the experimental data.

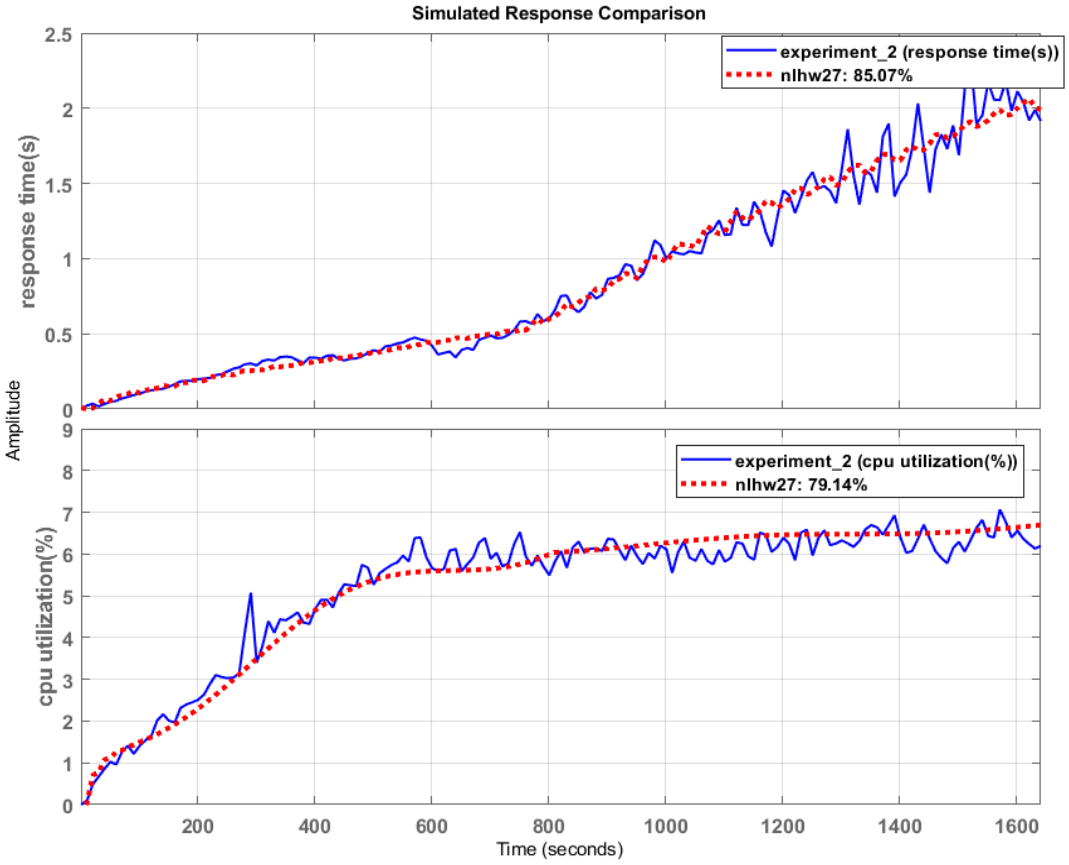

FIGURE 3.10. Estimated vs collected output. The output is collected from the experiment 2. The model has a response time fit percent of 85.07% and cpu fit percent of 80%

3.3. Conclusion

A non-linear multi-input multi-output model for the network has been identified using the multi-input multi-output system identification approach. The model and the fit percent accuracy demonstrates that the Hammerstein-Wiener model is suitable for the network irrespective of change in the network parameters and various types of inputs. Results convince that the derived model provides an average model fit accuracy of 87%. We consider the linear regions of the network and identify a linear model to capture the dynamics of the network. I present the identification of the network using a linear model in the next chapter.

TABLE 3.1. Goodness of fit with the predicted model

System	**Response time fit %**	**CPU utilization fit %**
10 Node	85.02	84.72
2 Node	85.07	79.14
Fast Arrival rate	83.22	81.12
Slow arrival rate	90.70	65.23
Medium arrival rate	84.83	73
Staircase Step increase	91	62

CHAPTER 4

FEEDBACK CONTROL FOR RESILIENCY IN THE FACE OF AN ATTACK

4.1. 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 [**17**]. 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 [**11**]. 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 on 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 are not able to adequately 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 [**52**] 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

This chapter is presented in its entirety from Vempati, Jagannadh, Mark Thompson, and Ram Dantu. "Feedback control for resiliency in face of an attack." In Proceedings of the 12th Annual Conference on Cyber and Information Security Research, p. 17. ACM, 2017 with permission from ACM.

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.

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

4.1.1. Related Work

Control theory approach has been applied to a wide range of computing systems [**72**]. There has been significant amount of research on identifying and performance modeling of web servers [**60**], [**100**], [**87**], [**92**]. Most of these models are based on linear time-invariant (LTI). There have also been several approaches for designing software systems using control theory [**63**]. An LPV approach to performance modeling of web server on a private cloud was presented in [**95**]. In this work, an operating region for 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 [**87**]. T Patikirikoral et al. proposed a Hammerstein-Weiner nonlinear model-based control for quality of service(QoS) management [**89**]. A control oriented model for performance management in a virtual environment was

suggested by Dharma Aryani et al., [**42**]. 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 [**52**]. These mechanisms do not model the attack.

To the best of our knowledge, the use of feedback control to build a 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 [**39**]. But their focus was on building efficient solution algorithms. The solution was a theoretical approach.

4.1.2. Motivation

DDoS attacks are evolving, and hackers are unleashing new techniques to amplify the attack. The financial impact of these attacks is growing. The average cost of a DDoS attack on an enterprise is over $2M per attack and is rising dramatically [**85**]. These attacks not only impact the financial costs but also damage the reputation of the organizations. With attacks increasing rapidly, the design of resilient systems is of utmost importance. Resilience is the ability of a system to withstand when provoked by an external disruption. Disruption or disturbance is an abnormal activity that hinders the normalcy of the system. Resilient networks try to provide the desired level of service, despite challenges such as malicious attacks, and misconfigurations.

In our previous work, we developed a robust closed-loop feedback mechanism to maintain the stability of a web service in the face of an attack [**108**]. We presented a passive approach of distributing the load to multiple links and reducing the impact of an attack, using feedback control. The results proved that the proposed feedback mechanism provided a positive effect on the system and the web services were restored in real time. The impact of the attack was reduced in about 60 seconds, and the user quality of experience was maintained, making the network stable.

Our work focuses on designing a robust closed-loop feedback control mechanism to protect the network and provide resilience to a network when triggered by attacks and faults. We develop this feedback control mechanism for real-time traffic, such as streaming video

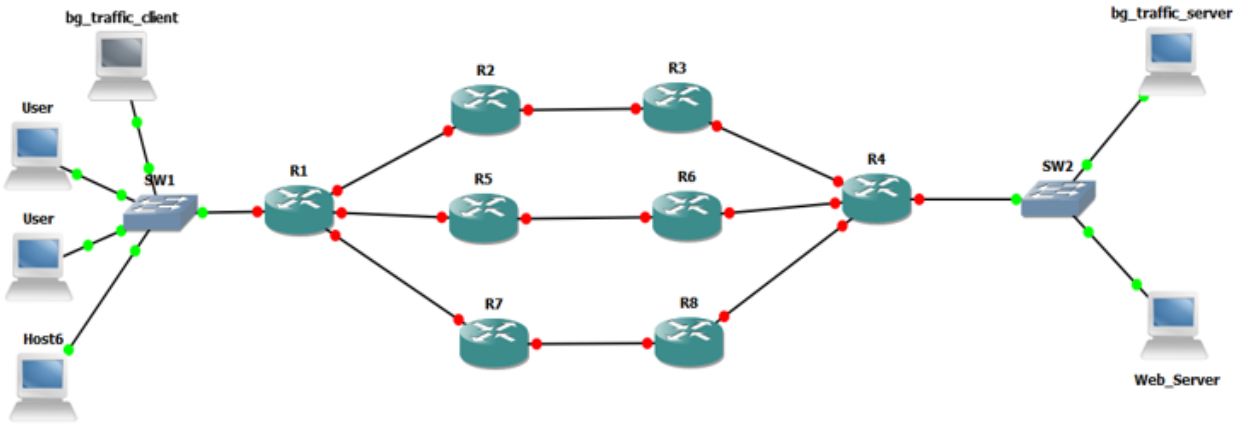

FIGURE 4.1. Experimental network topology: network elements connected in parallel. The routers are configured with a round robin load balancing feature to share the load equally among the links.

and audio, which is very sensitive to delay, packet loss, and jitter. Slight disruptions in the network can hinder the performance of the real-time traffic. Network traffic such as the real-time traffic is highly dynamic, complex and very unpredictable. Understanding this complex nature of the network traffic is critical in being able to design an accurate model. In our approach, we look to system identification to build and validate a model for the complex and dynamic network.

Current mitigation techniques ranging from hours to days are entirely unacceptable given the cost and inconvenience these attacks place on our society. Our mechanism provides a real-time and scalable solution to maintain the service level agreements and to deliver video and audio streaming seamlessly smooth in spite of an attack. After designing the dynamic feedback controller, the question arises can we quantify resilience, if so, how? We look to control system theory to define the metrics of the resilient system.

4.2. Architecture

4.2.1. 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 4.1. The network

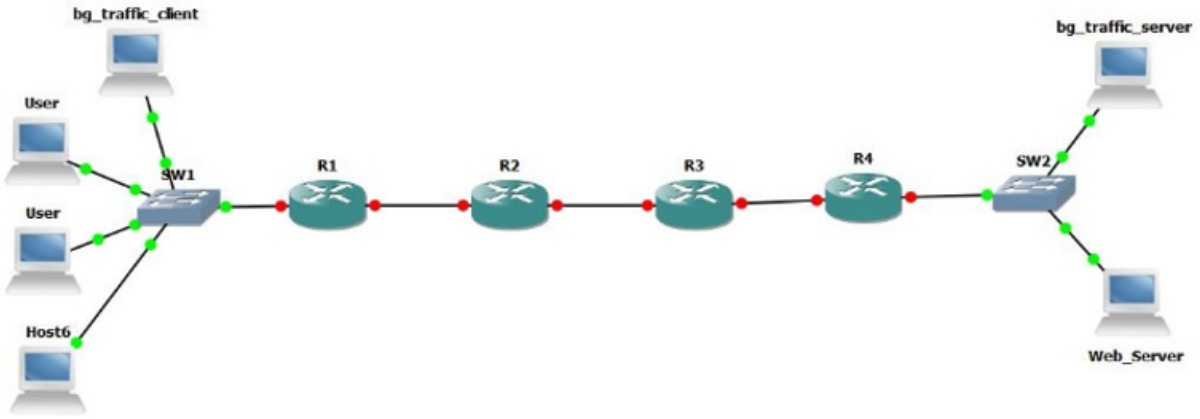

FIGURE 4.2. Experimental network topology: network elements connected in series.

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 4.1 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 to be able to measure the performance of our network relative to one connected in series as shown in Figure 4.2. In this manner, we can 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.

4.2.2. Client

Linux containers (LXC) [**16**] were used to emulate real-world scenario. LXC is an operating system level virtualization that can 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 the same subnet. In this manner, we can model a significant number of users making requests to the server.

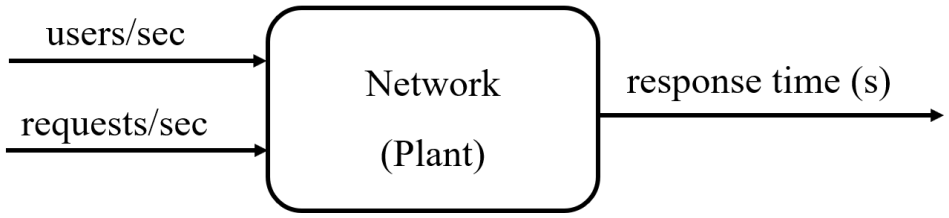

FIGURE 4.3. Open loop system. The input variabes are number of users and server hits; The output variable measured is response time to access the web page.

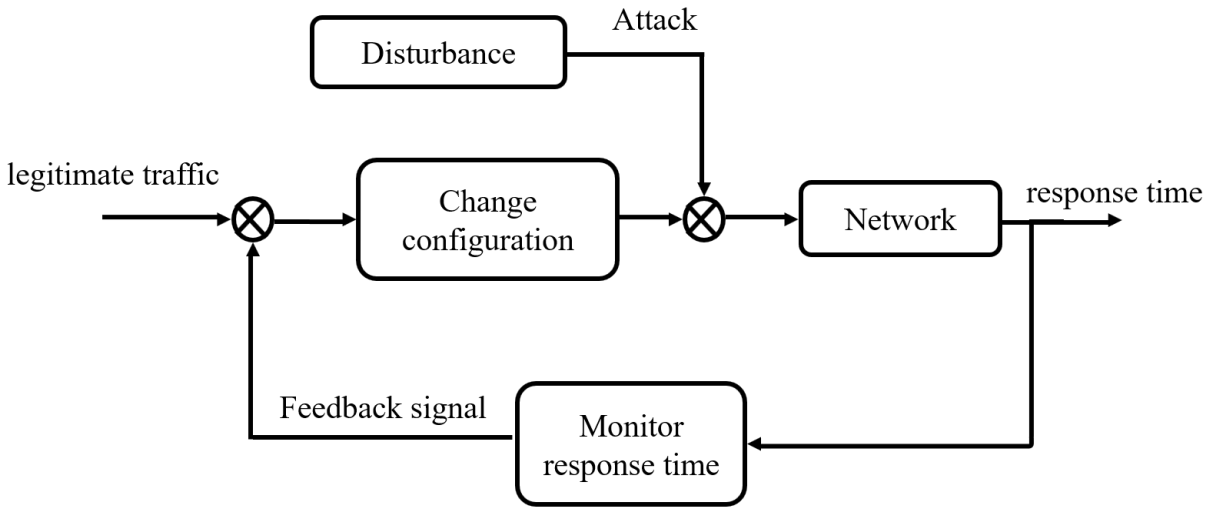

FIGURE 4.4. Automatic feedback control: Real-time configuration changes in the event of a Disturbance (e.g., DDoS attack)

4.2.3. Server

A standalone machine is used to host an Apache HTTP web server [**35**] 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.

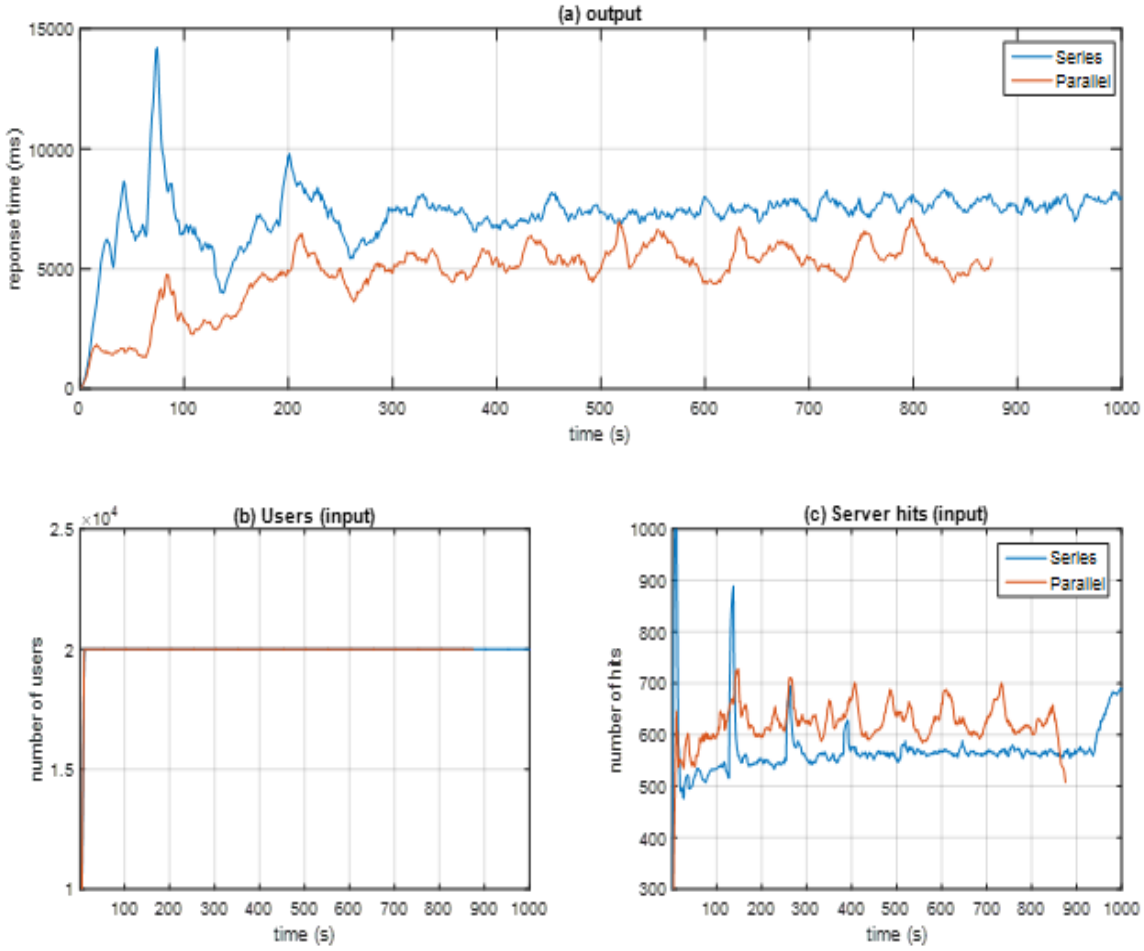

FIGURE 4.5. System response for the network connected in series and in parallel topologies. (a) Response time for the HTTPS requests for the series and parallel network configuration (b) and (c) are the users and number of hits connected in series and parallel.

4.3. Methodology

4.3.1. System Identification

To be able to analyze the stability and sustainability of a network 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 [**80**]. Using this process, we are able to obtain a mathematical relationship between the input and output data.

We use a black box approach to identify the model of the plant as shown in Figure 4.3. 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 [**28**]. 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.

4.3.2. 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.

4.3.3. 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 [**1**] 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.

4.3.4. Background Traffic

We use the hping3 [**15**] tool as a background traffic generator to insert homogenous traffic along the routers R1 and R3 as shown in Figure 4.1. This tool can generate this background traffic when the network is connected in a series or when the parallel links are enabled, distributing traffic evenly among the three active paths.

4.3.5. Feedback

Prior to being able to implement the feedback architecture shown in Figure 4.4, 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 4.4. In this study, we consider the homogeneous background traffic to be legitimate traffic. The response time is fed back to the proportionality controller, which is aware of

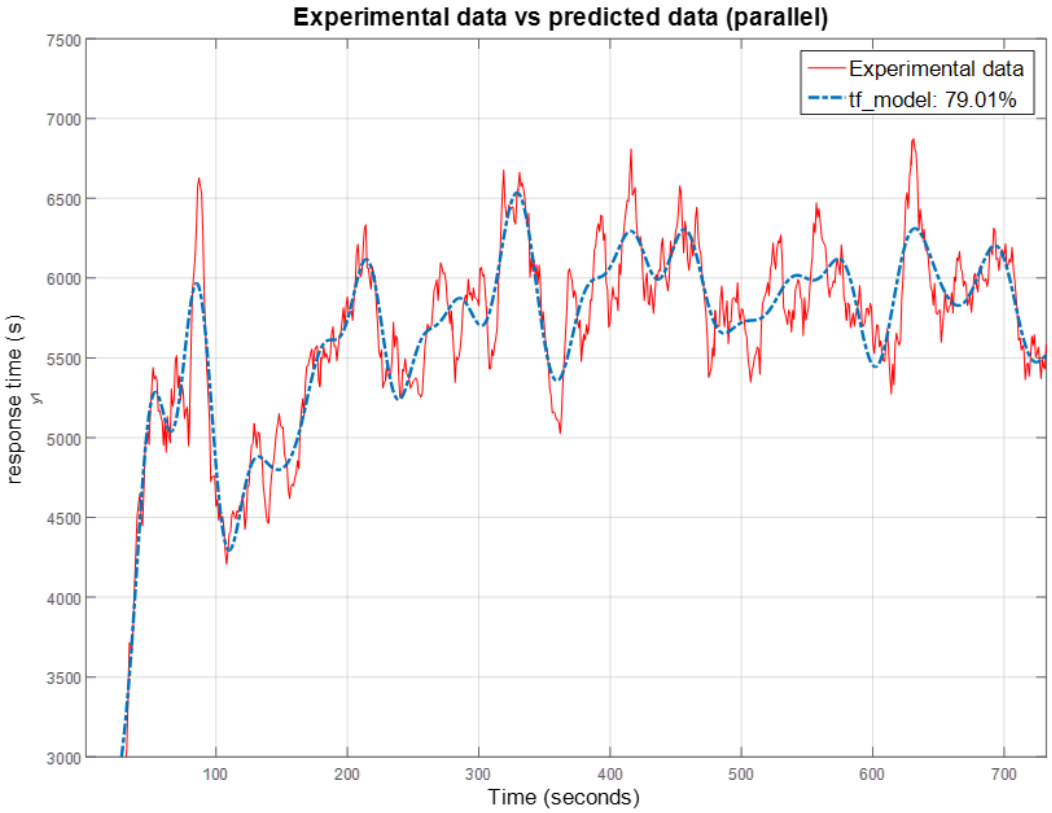

FIGURE 4.6. The transfer function (through system identification) closely follows the experimental data even during an attack (observed 79% match). Parallel network topology was used as shown in Figure 4.1

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.

4.4. 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 during an attack. Figure 4.5 shows this input-output of attack traffic with background traffic data collected from the network connected in series and parallel.

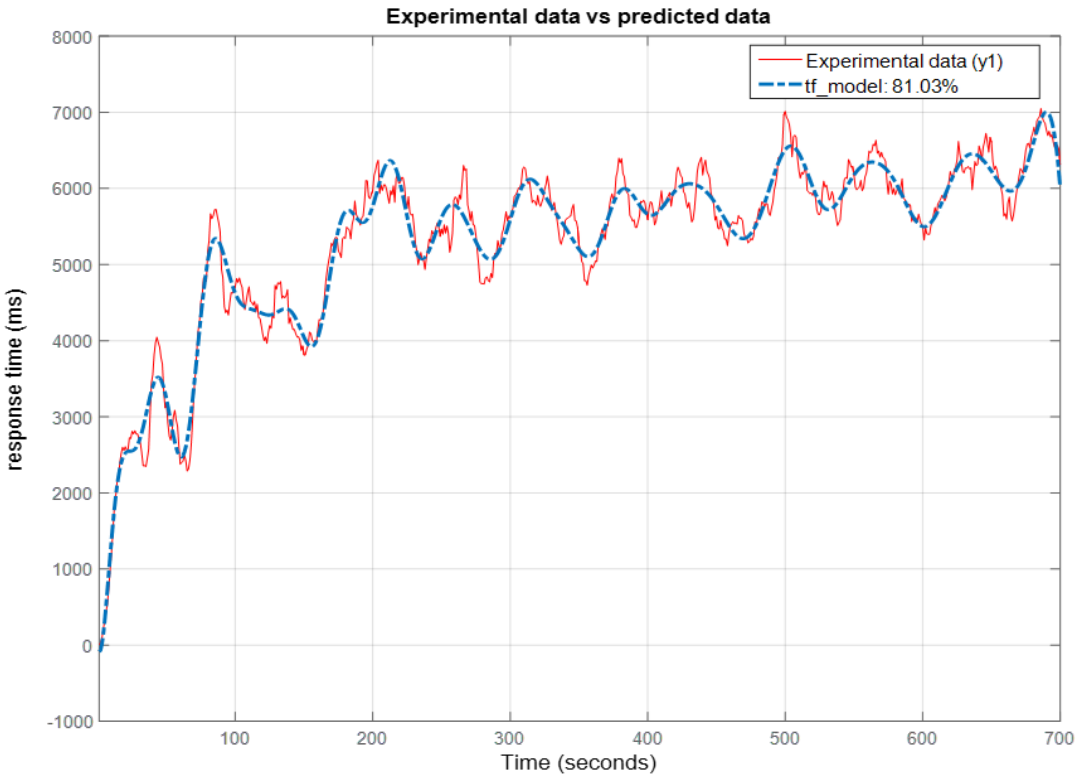

FIGURE 4.7. The identified transfer function model closely follows the data collected from the network under attack connected in series without the presence of background traffic.

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.

As is shown, the network is loaded with 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.

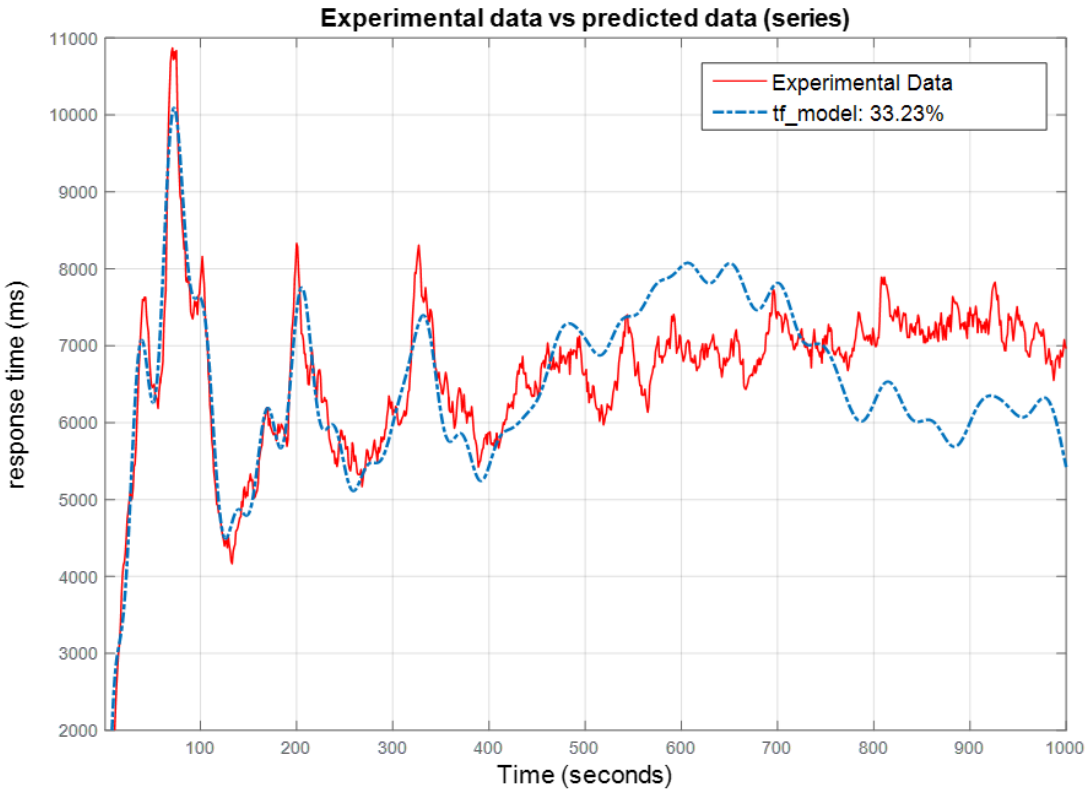

FIGURE 4.8. The experimental data does not follow the identified transfer function when the network is under attack. This poor fit % indicates that the network is unstable.

4.4.1. Stability Analysis

4.4.1.1. 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 4.7 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.

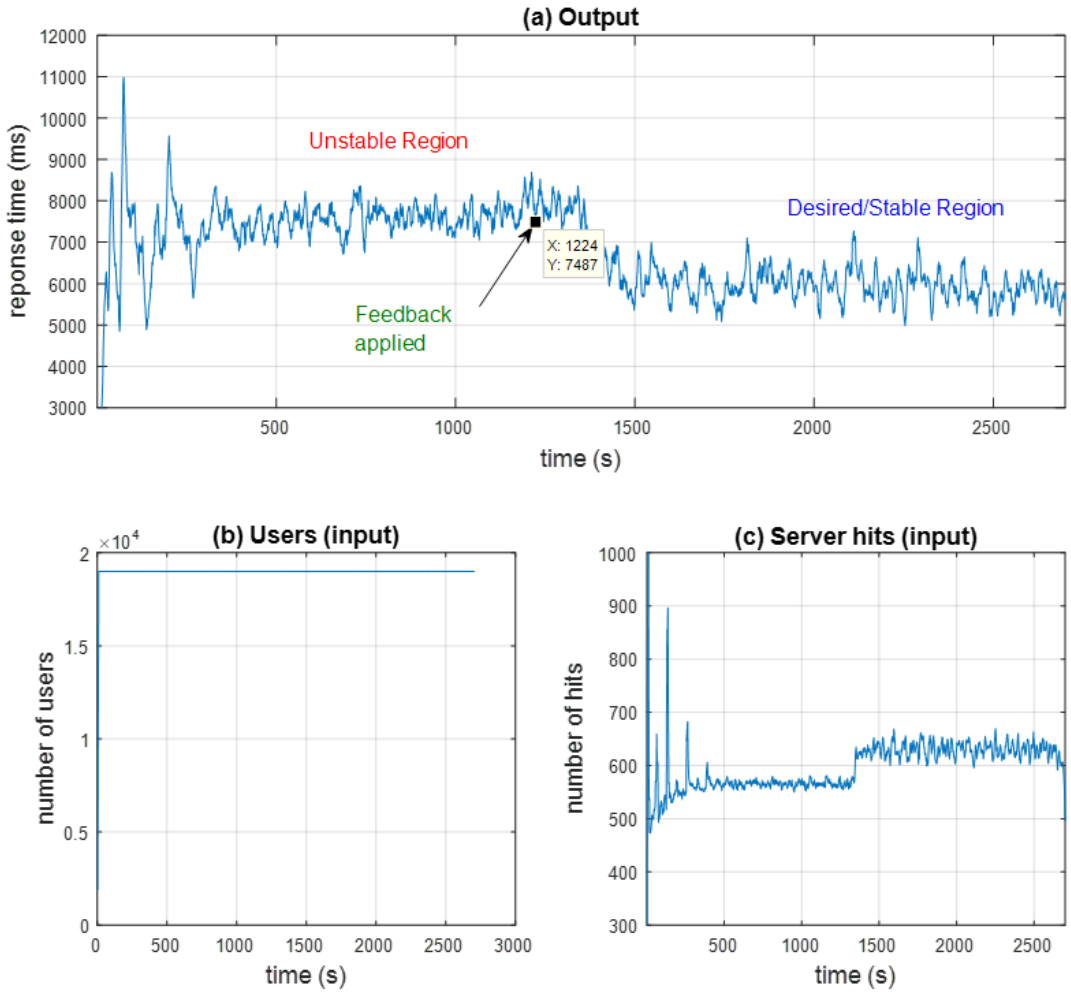

FIGURE 4.9. (a) Response time of the network before and after the attack . (b) and (c) are the inputs provided to the network. The feedback is applied at time = 1224 seconds to the network under attack. The network goes to stable region from unstable state within 300 seconds. (c) shows the increased number of hits due to the increase in throughput.

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 4.6. We observe that the fit accuracy for our network connected in parallel is nearly 80%, which now serves as our benchmark for us to

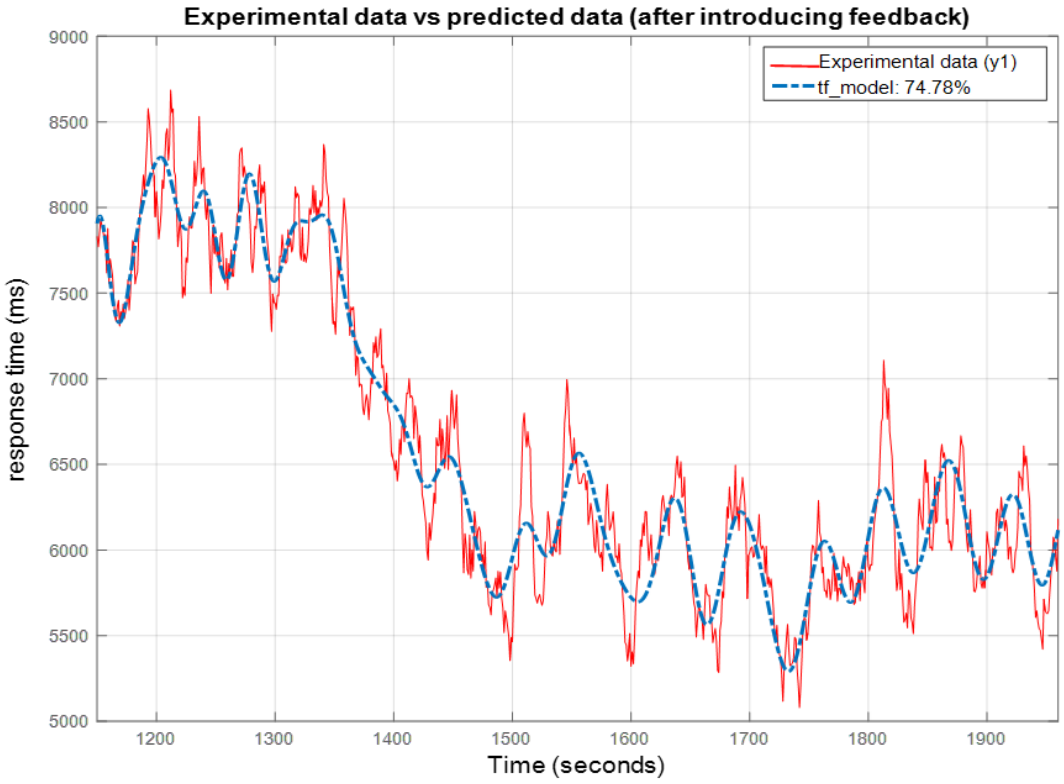

FIGURE 4.10. Extension of Figure 4.9; The transfer function model closely follows the data after the feedback is introduced. This proves that the network goes to stable state from unstable state after feedback is applied

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 the 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.

4.4.1.2. Unstable system

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 4.8, 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.

4.4.2. 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 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, remains steady throughout the attack.

What is essential in Figure 4.9 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 4.10 shows the stable region of Figure 4.9 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.

4.5. Conclusion

The attack traffic is modeled in this approach to ensure the network maintains a stable state with any 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

TABLE 4.1. This table summarizes the average response time, when the network is connected both in series and parallel.

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

TABLE 4.2. This table shows the accuracy of the predicted data against the experimental data, when the network is connected both in series and parallel

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

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 better fit accuracy. When the data is filtered further, lower order models yield better accuracy; however, we might lose critical data.

4.5.1. Future Work

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 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.

CHAPTER 5

UNINTERRUPTED VIDEO SURVEILLANCE IN THE FACE OF AN ATTACK

5.1. Introduction

Distributed denial of service (DDoS) attacks continue to be problematic for websites and service providers and thanks to the Internet of Things (IoT) supporting a rapidly growing number of network-connected devices from refrigerators to thermostats, and so on, this trend is expected to continue [**30**]. During the second quarter of 2017, for example, the number of DDoS attacks increased by 28 percent, with the United States being targeted over 122.4 million times [**82**]. One high profile DDoS attack during this period involved Microsoft's instant messaging service Skype where many users lost connectivity to the application and were unable to send or receive messages for several days with lingering connectivity issues. In response to this attack, Stephanie Weagle, VP of Corero Network Security, stated that "proactive, automated protection is required to keep the internet-connected business available in the face of DDoS attacks" [**86**]. In order to mitigate these growing DDoS attacks, we need to be able to identify and respond to malicious traffic immediately.

5.1.1. Motivation

Mitigating the effects of a DDoS attack in a few hours, much less in a few days, is unacceptable given the financial impact on businesses and consumers every DDoS attack has. The average damage of a single DDoS attack on business has now increased to more than $2.5 million per incident [**85**] while the cost to launch a DDoS attack ranges from a measly $2,000 to $7,275 [**51**]. What's more is that DDoS attacks are quickly evolving and taking a life of their own as they are growing larger and more complex than ever. In addition, there is a burgeoning market for DDoS-as-a-Service as the sales of botnets, and DDoS tools have

This chapter is presented in its entirety from Vempati, Jagannadh, Ram Dantu, and Mark Thompson. "Uninterrupted Video Surveillance in the Face of an Attack." In 2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE), pp. 843-848. IEEE, 2018 with permission from IEEE.

grown into a sizable business [**30**].

In our previous work, we developed a robust feedback design to maintain the stability of the network despite attacks using non-real-time (NRT) traffic [**108**]. We presented a passive approach to minimize the impact of DDoS on web services. As evidenced by the results, the feedback mechanism provided a positive effect on the network and stabilized the network in less than 60 seconds. The results in [**108**] also show a fit accuracy of approximately 75% after the feedback was provided, bringing back the unstable network to a stable state.

In this paper, we consider real-time traffic, such as streaming video, which is very sensitive to delay, packet loss, and jitter. Slight disruptions in the traffic can deteriorate streaming video. Understanding this unclear nature of real-time traffic and generating a corresponding model can be very challenging. A simple model is not adequate to identify the dynamics of such traffic. We are thus motivated to look to system identification [**80**] techniques to design and analyze a robust model. We select autoregressive-moving average with exogenous terms model to identify the network. We then validate the model with our robust feedback strategies. We use a micro-firewall rule that identifies and prioritizes the legitimate traffic. The firewall rule acts as a feedback controller that is used to detect anomalies in the quality of the video. The importance of this mechanism is that it provides a real-time and scalable solution to deliver seamless video streaming in spite of an attack.

5.1.2. Related Work

A feedback control approach has been applied to a wide range of network systems [**24**]. There has been a significant amount of research on performance modeling of video streaming [**55**–**57**, **77**, **103**, **113**]. A feedback control architecture was designed by Luca De Cicco et al., [**58**] for adaptive live video streaming. They propose a quality adaption controller for live adaptive video streaming. In their proposed design, the controller is fed with the length of the sender buffer as input. The sender buffer selects the video quality. The authors claim that their architecture is able to control the video level with a transient time of 30 seconds.

Chao Chen et al., [**49**] presented a Hammerstein-Wiener model for predicting the time-varying subjective quality (TVSQ) of rate-adaptive videos transmitted over HTTP.

The authors claim that their model predicts TVSQ for the HTTP-based video streaming in real time. They also claim that their model achieves an outage rate of less than 3.4%. A non-linear autoregressive model with exogenous outputs model was proposed for the prediction of streaming video quality of experience (QoE) in [**45**]. Their model is driven by three inputs – objective measure of video quality, rebuffering-aware information and QoE memory descriptor.

Guibin Tian and Yong Liu [**103**] developed a video adaption algorithm for Dynamic Adaptive Streaming over HTTP (DASH). Their algorithms use client-side buffered video time as the feedback signal. They use a PI controller driven by deviation in buffered video time as the feedback signal.

These methods focus on maintaining the quality of the video at the server-side or the client-side. To the best of our knowledge, the use of feedback control to mitigate the effect of denial of service (DoS) attack has to be yet explored for real-time traffic. Our approach has a solid defense mechanism. The feedback mechanism implemented in our approach stabilizes the network and makes the network function robustly despite the attack. Also, we prove that the QoS of the video remains the same even after the network is disrupted.

5.2. Architecture

5.2.1. Network Topology

We implement the same network topology used in [**108**]. Specifically, we implement a network connected in parallel using eight Cisco Catalyst devices configured to use the Open Shortest Path First (OSPF) protocol as shown in Figure 5.1. When our feedback mechanism is applied, the full network of three parallel links, each comprised of four routers connected in series, become enabled to respond to disruptions or failures in the network. Each link is configured to share the load equally, with the F1 router utilizing the load-balance feature to distribute the packets based on the destination address.

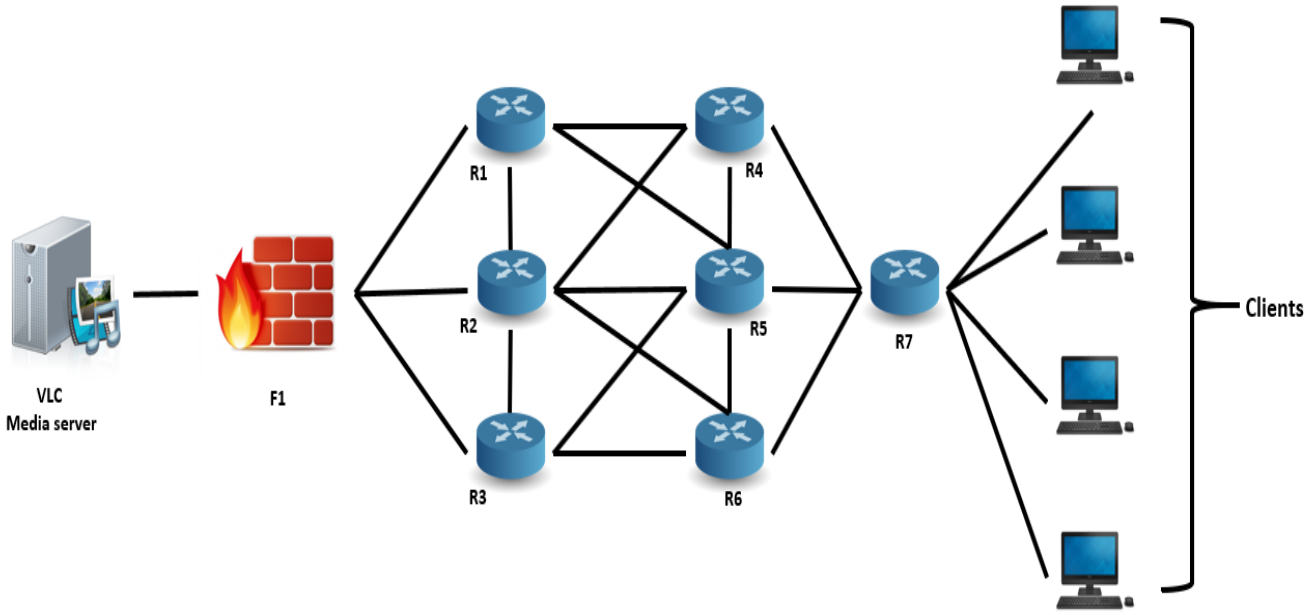

FIGURE 5.1. Experimental network topology: network elements connected in a mesh topology. All the network devices (F1: firewall, and R1-R7: routers) are configured with open shortest path first (OSPF) routing protocol to route the packets. The routers are configured with a round robin load balancing feature to share the load equally among the links.

5.2.2. Client

We use the VideoLAN client (VLC) media player [**34**] to view the streaming videos. The streaming server's network IP is entered as the network URL. Multiple clients installed with VLC media player were used to emulate real-world scenarios.

5.2.3. Server

A standalone machine is used to host a video-streaming server to deliver real-time video content over the Internet to a user with a connected device. This system uses an Intel Xenon processor (4 cores) with 32 GB RAM. H.264/MPEG-4 AVC is used as the video coding format.

5.3. Methodology

5.3.1. System Identification

We use a black-box approach to identify and analyze the dynamics of the network. We consider the entire network made up of clients, a streaming server, and network devices such as routers, switches, etc. We then model this network using the system identification approach. This model describes the relationship between the measured input and output.

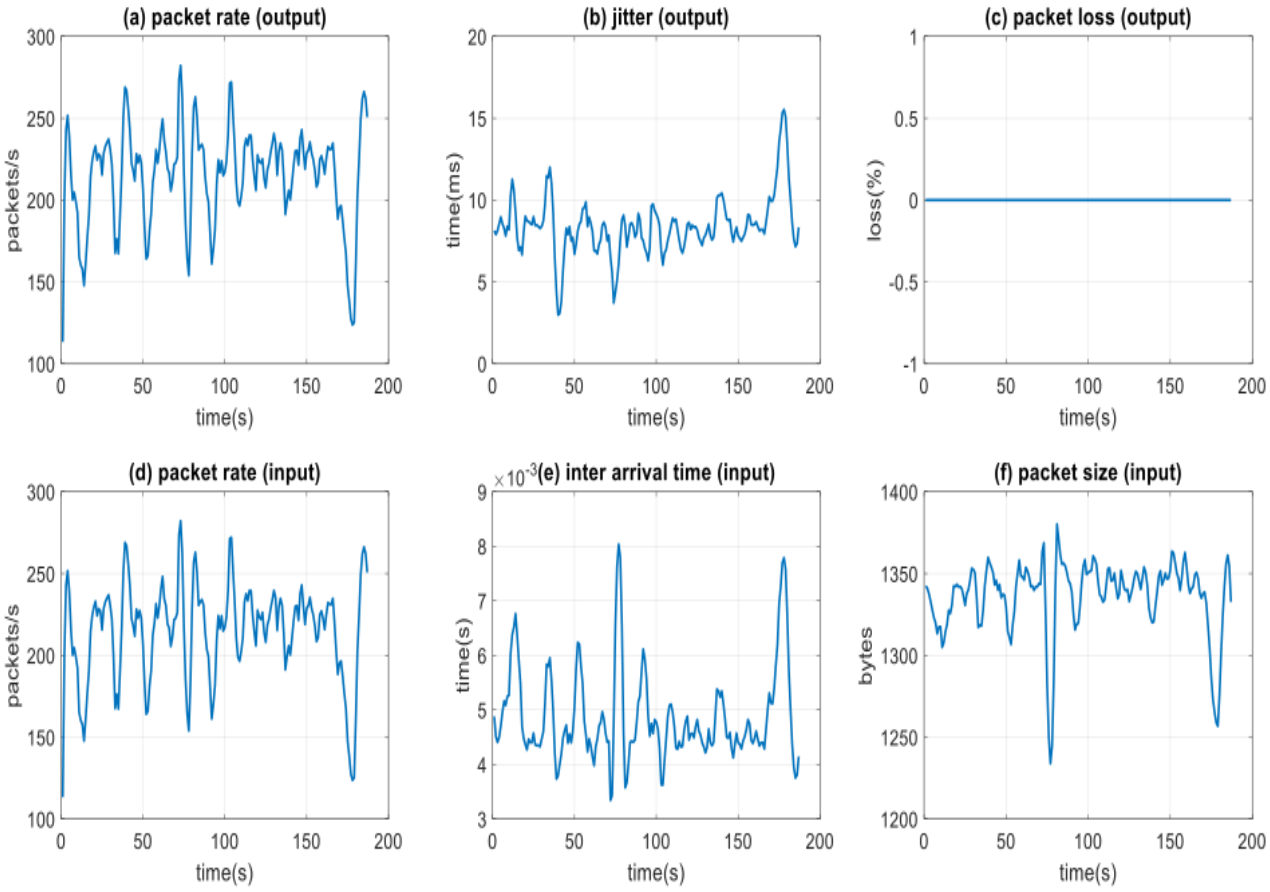

FIGURE 5.2. Input-output profile. This data is used to model the network. Figure (d), (e) (f) are input to the network and (a), (b) and (c) are the outputs collected from the network

Real-time traffic, such as video streaming, is very sensitive to jitter, delay, and packet loss. The inter-arrival time and the rate of incoming packets directly affect the QoS metrics. Hence, we select the average inter-arrival time and the rate of packets measured from the

streaming server as inputs. The input is represented as u(t). The outputs of the system, represented as y(t), are the QoS metrics such as end-to-end jitter, packet loss, and the rate of RTP packets between the client-streaming server pair.

This Multi-Input Multi-Output (MIMO) black-box model is described using the System Identification Toolbox present in Matlab [**28**], which constructs an analytical model of the dynamic network from the observed data. This system identification technique is widely used in control engineering.

We select an autoregressive moving average with exogenous terms (ARMAX) model structure to identify the black-box model. ARMAX models are more flexible in handling disturbances [**24**] and are encouraged to use for time series modeling. This model structure is useful when load disturbances are present in the input.

For parameter estimation of the ARMAX model, we initially collect the input-output data for a sampling period of one second, as shown in Figure 5.2. This data is split into two components; the first half used to generate the model and last to validate the model. We then use the ARMAX model order range and estimate the parameters using the System Identification Toolbox. Using the fit criteria, we select the best model and then validate it using a different set of the data sample. The order of the model is selected based on the fit accuracy.

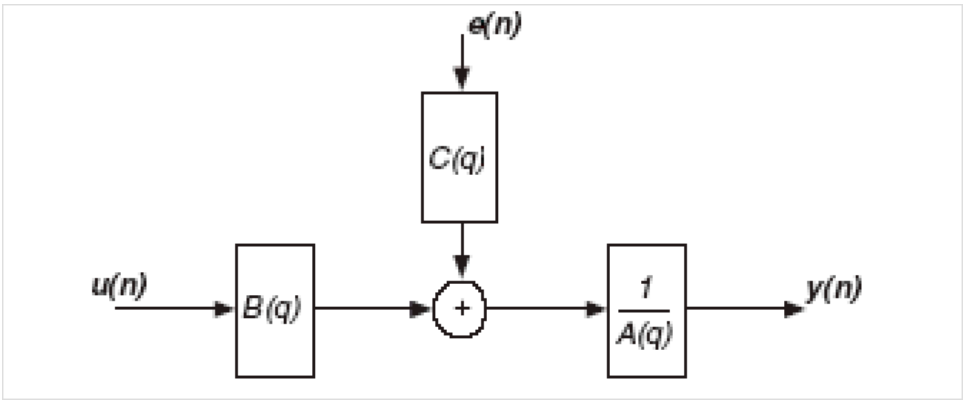

FIGURE 5.3. ARMAX general architecture

The general difference equation of ARMAX structure is:

$$A(q)y(t) = B(q)u(t - nk) + C(q)e(t) \tag{2}$$

where,

y(t)— Output at time t. The orders of the A, B, and C polynomials are na, nb, and nc respectively.

n_a - Number of poles.

n_b - Number of zeroes plus 1.

n_c - Number of C coefficients.

n_k - Number of input samples that occur before the input affects the output, also called the dead time in the system.

u(t - n_k) ... u(t - n_k - n_b + 1) - Previous and delayed inputs on which the current output depends.

e(t - 1) ... e(t - n_c) - White-noise disturbance value.

The parameters n_a, n_b, and n_c are the orders of the ARMAX model, and n_k is the delay. Q is the delay operator. Using the ARMAX model jitter, the rate of RTP packets and packet loss can be predicted in the near future. The jitter and the arrival rate could be disturbed by the unpredictable cross traffic, making it difficult to predict in the long term. The ARMAX model generated is:

Model for output "Rate":

$A(z)y_1(\mathrm{t}) = -\ A_i(\mathrm{z})y_i(\mathrm{t}) + B(z)u(t) + C(z)e_1(\mathrm{t})$

$A(z) = 1 + 0.7408z^{-1} + 0.0006638z^{-2}$

$A_2(\mathrm{z}) = 0.01649z^{-1} - 0.01265z^{-2}$

$A_3(\mathrm{z}) = 0$

$B_1(\mathrm{z}) = 0.9892 + 0.7505z^{-1}$

$B_2(\mathrm{z}) = -201.8 + 176.7z^{-1}$

$B_3(\mathrm{z}) = -0.005782 + 0.006175z^{-1}$

$C(z) = 1 + 0.8317z^{-1}$

Model for output "Jitter":

$A(z)y_2(\mathrm{t})$ = - $A_i(\mathrm{z})y_i(\mathrm{t})$ + $B(z)u(t)$ + $C(z)e_2(\mathrm{t})$

$A(z) = 1 - 1.613z^{-1} + 0.7593z^{-2}$

$A_1(\mathrm{z}) = 0.01649\ z^{-1} - 0.01265\ z^{-2}$

$A_3(\mathrm{z}) = -7.007e{-}25\ z^{-1} - 2.38e{-}25\ z^{-2}$

$B_1(\mathrm{z}) = -0.03476 - 0.04866z^{-1}$

$B_2(\mathrm{z}) = 681.9 - 704.9z^{-1}$

$B_3(\mathrm{z}) = 0.03101 - 0.02925z^{-1}$

$C(z) = 1 + 0.03994z^{-1}$

The orders of n_a, n_b, n_c, and n_k are 2, 2, 1, and 0, respectively. The mean square error of the model is 0.8733. The model yielded a fit accuracy of 98% and 78% for the rate of packets and jitter outputs, respectively.

5.3.2. Generation of Traffic

We use VLC media player to broadcast a stream. We select the Real Time Streaming Protocol (RTSP) as the streaming method with the H.264 video compression codec and MP4 container format. The selected video was streamed across the network shown in Figure 5.1 The tcpdump packet analyzer is used to sniff the packets at the server's interface, thus acting as the sensor. The data such as rate of packets, inter-arrival time, etc. is collected periodically from the packet capture. We use this collected data as input into our model.

5.3.3. Feedback

We implement and analyze three different feedback mechanisms

5.3.3.1. DSCP Markdown

Differential Services Code Point (DSCP) is simply a measure of the QoS level of a packet. As a passive approach, this rule still allows the attack traffic but lowers the priority of the out-of-profile packets by marking them with a different QoS level and prioritizes the real-time in-profile packets.

5.3.3.2. Drop Out-of-Profile Traffic

As an active approach, this rule drops out-of-profile traffic based on the rate and burst size parameters. The rate defines the number of packets removed at each fixed 0.125 milliseconds interval while the burst size is the maximum number of packets that can be held by the bucket to determine whether a packet is in profile or out-of-profile.

5.3.3.3. Parallel Links

In this passive approach, the attack traffic is still allowed, but parallel links provide additional bandwidth to stabilize the network when an attack is detected.

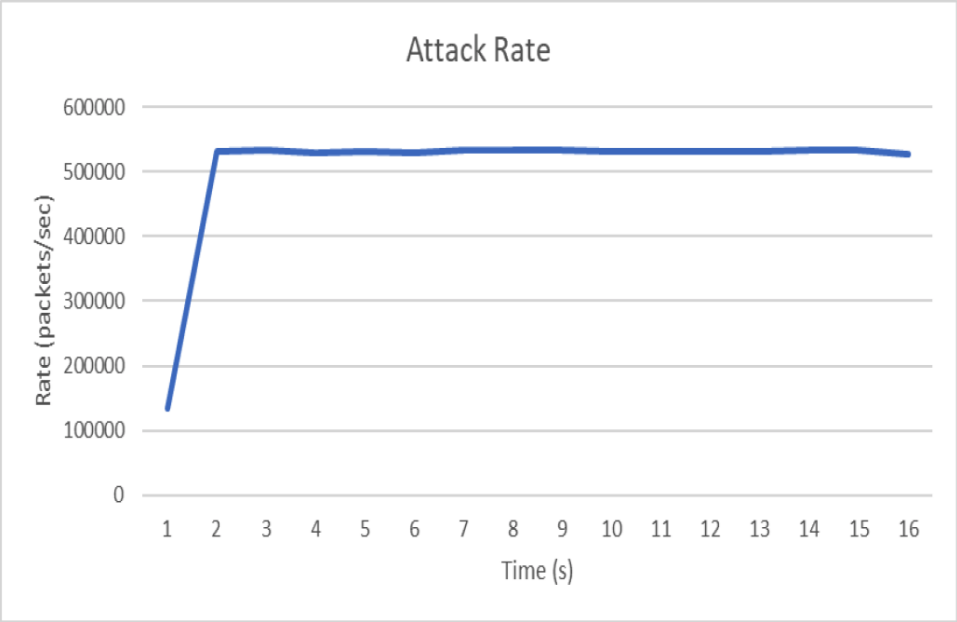

FIGURE 5.4. Arrival rate of the DoS attack. This rate resembles a step input

5.4. Analysis of Results

The feedback to the network is applied by adding a micro-firewall rule that contains two parameters rate and burst size that control the operation of policing. These parameters are selected based on the measured traffic rate. Two types of policing actions can be performed if the traffic complies with the specified profile. They are (i) dropping the out-of-profile packet and (ii) marking down the DSCP value of the packet to the one with a lower priority.

According to Cisco [**32**], QoS policing in the Catalyst 3550 device complies with the leaky bucket concept to determine whether a packet is considered in-profile or out-of-profile. That is, if there are enough tokens available for a packet to be transmitted, it is considered to be in-profile; otherwise, it is considered to be out-of-profile. In effect, the number of packets proportional to the traffic packet sizes is placed in a bucket so that at regular intervals, the tokens derived from the configured rate are then removed. If there is no place in the bucket to accommodate a packet, the packet is considered as out-of-profile and is dropped or marked down. These actions were applied dynamically to the network as a feedback mechanism when the network moves to an unstable state.

5.4.1. Marking Down DSCP

The feedback rule applied in this scenario changes the DSCP value of the out-of-profile traffic to a lower priority and prioritizes the real-time traffic. Figure 5.6 shows the output collected from the network before and after the attack. The network is attacked after 85 seconds. The rate of attack is approximately 530,000 packets per second as shown in Figure 5.4 that resembles a step input. The attack is the disturbance to the plant. The comparison of the 1-step predicted data with the experimental data when the network under attack is shown in Figure 5.5

From Figure 5.6, we can observe that the jitter measured has erratic spikes and is around 38 ms. This jitter value causes severe deterioration in the quality of the video. Figure 5.6 also shows the measured packet loss and the rate of RTP packets before and after the attack, respectively. When the network is under attack, the nodes appear congested. As a result, several packets would be dropped, indicating that the network is unstable. We can observe from Figure 5.6 that 70 percent of the packets are dropped during the attack. Due to a very high drop in the percentage of the packets, we can observe the jitter increased by approximately 50 percent.

After applying the rule of marking down out-of-profile traffic and prioritizing the real-time traffic, the network returns to a stable region after being unstable. We can observe that, when feedback is applied to our network, the jitter dropped down to nearly 12 ms with

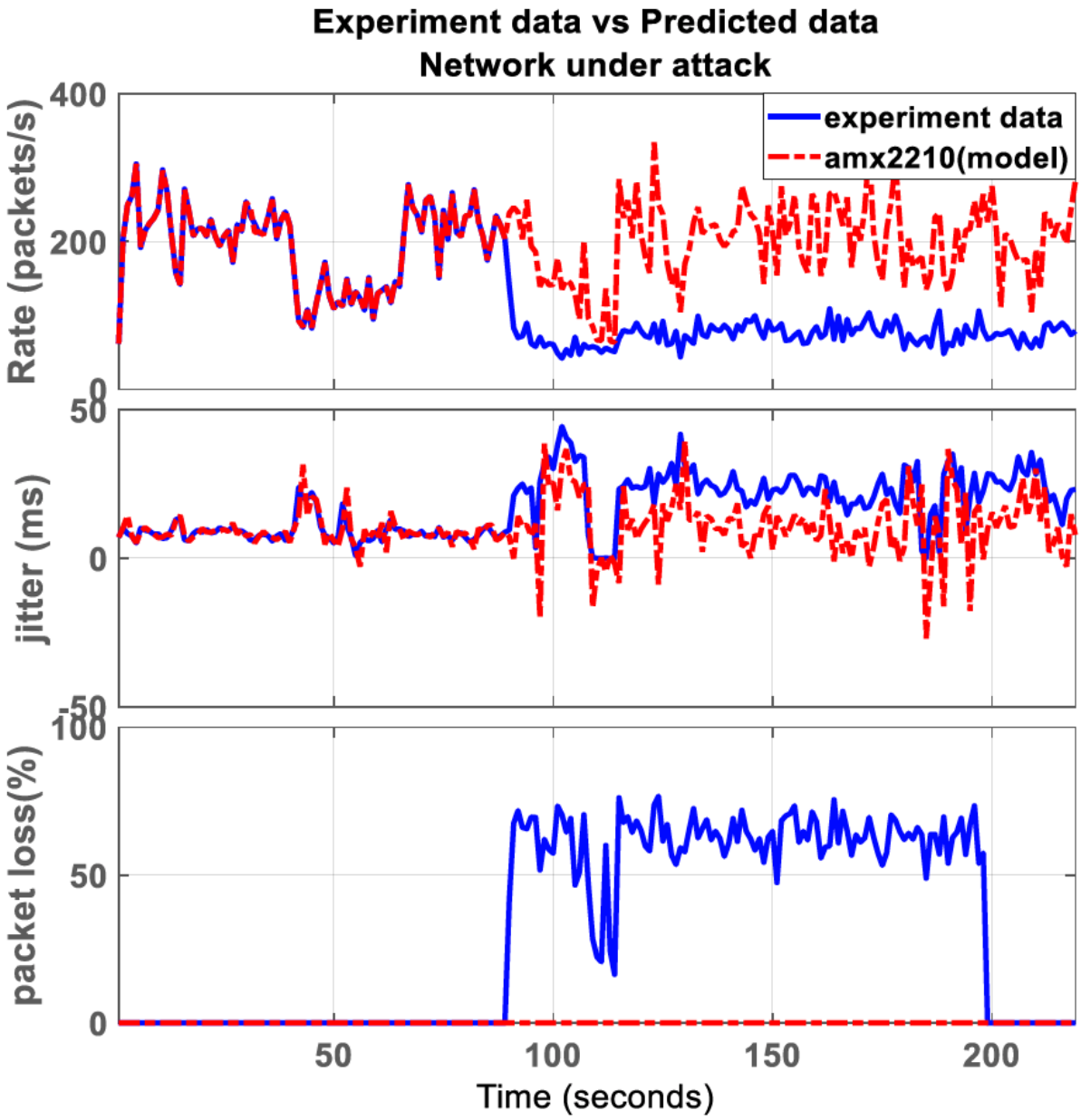

FIGURE 5.5. The experimental data does not follow the predicted data when the network is under attack. The network is attacked at t = 80s. Due to the attack, jitter and packet loss increase, resulting in the drop of the packet rate.

no packets lost. The feedback is applied at 145 seconds in the scenario. The network then goes from an unstable state to a stable state within 10 seconds.

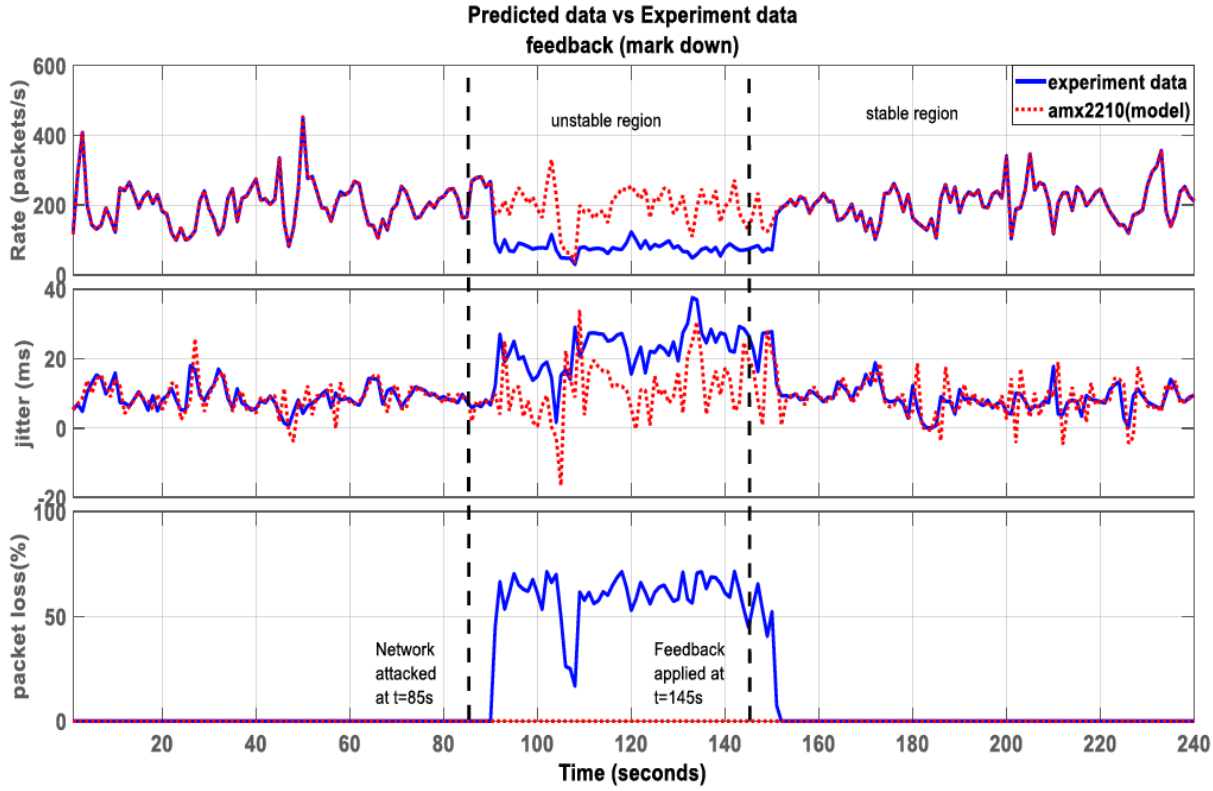

FIGURE 5.6. (a) rate of packets, (b) jitter and (c) packet loss measured from the RTP stream before and after the attack. The network is attacked at t=85s. The feedback marking-down rule is applied at t = 145s to the network under attack. We can observe, the model follows the data initially when the network is in a stable state. During the attack, the predicted response does not match with the experimental data. The model again closely follows the data after the feedback is introduced. This shows that the network goes to a stable state from the unstable state after the feedback is applied.

5.4.2. Dropping Out-of-Profile Traffic

The feedback rule applied in this scenario drops the out-of-profile traffic. Figure 5.7 shows the traffic profile of the network before and after the feedback is applied. The feedback applied to the network under attack drops the traffic that does not comply with the specified profile. The real-time traffic matches with the specified rule and gets placed into the bucket. The attack traffic with a very high rate and burst size gets dropped, reducing the congestion

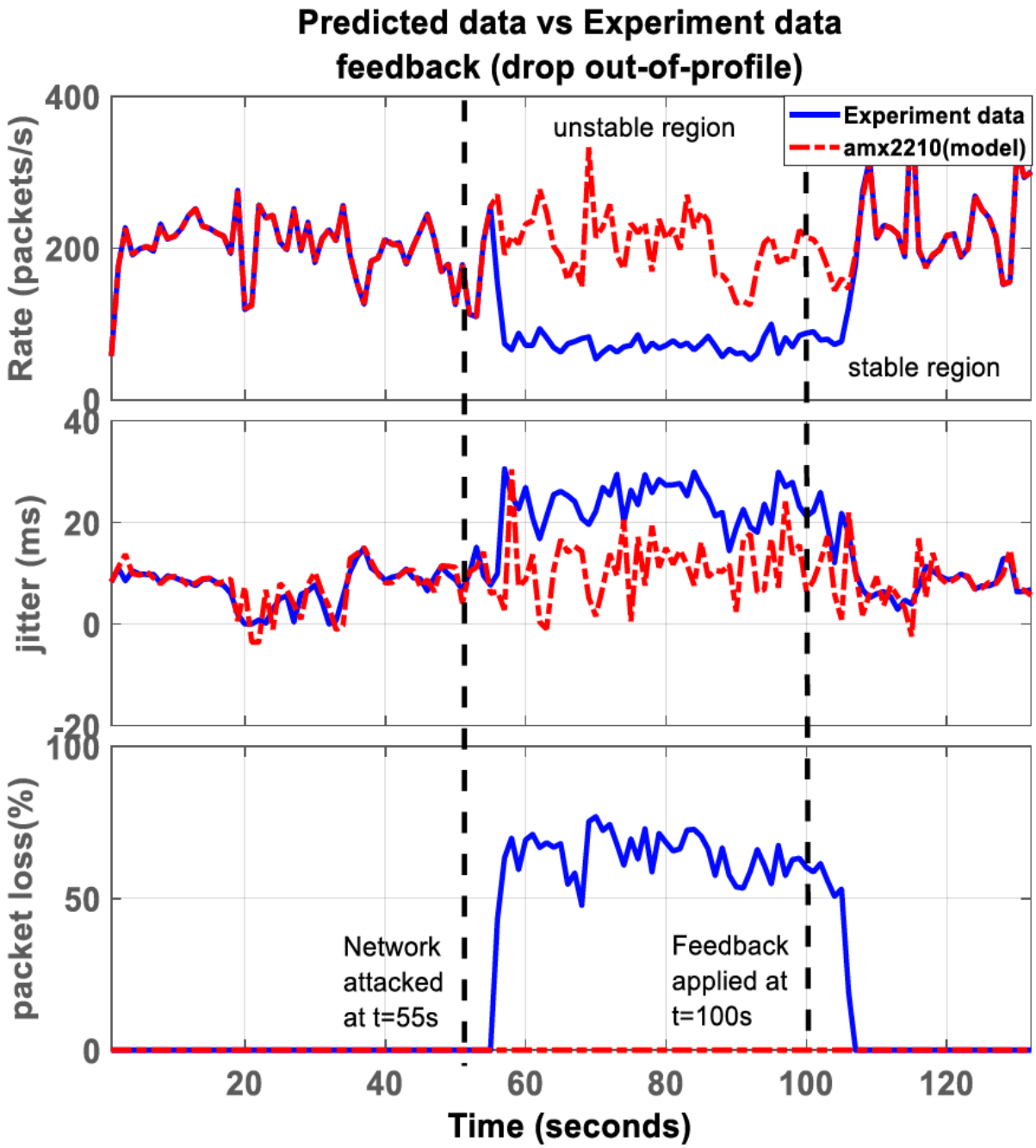

FIGURE 5.7. (a) rate of packets, (b) jitter and (c) packet loss measured from the RTP stream before and after the attack. The network was attacked at t=55s. The feedback applied which drops the traffic that does not comply with the rule is applied at t = 100s. The model closely follows the data after the feedback is introduced and yields a better fit. This shows that the network goes to a stable state from the unstable state after the feedback is applied.

at the router. The network is restored back to a stable state in less than 10 seconds.

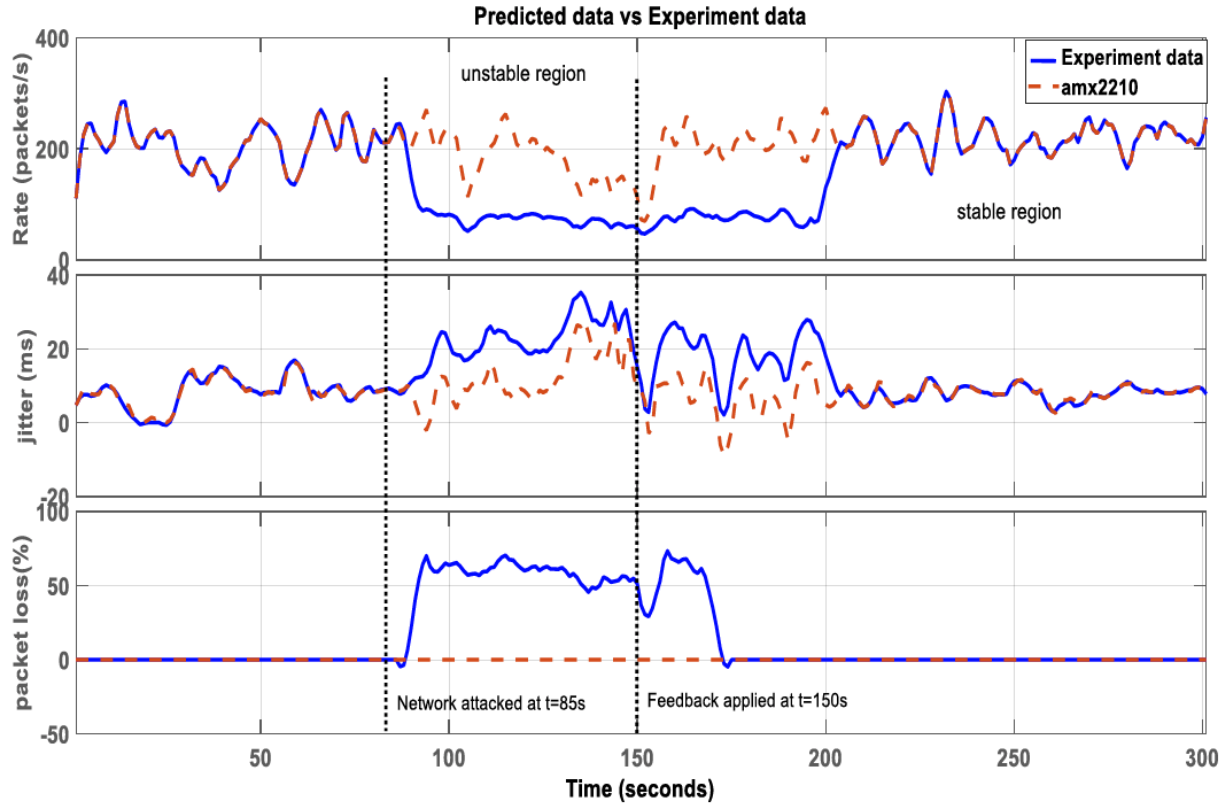

FIGURE 5.8. (a) rate of packets, (b) jitter and (c) packet loss measured from the RTP stream before and after the attack. The network is attacked at t=85. The feedback adding a parallel path is applied at t = 150s. We can observe that after the application of the feedback the network takes about 60s for the network to restore back to stable state

5.4.3. Parallel Path

Figure 5.8 shows the output traffic profile of our network before and after feedback is applied. With our network initially connected in series, feedback is applied to the network once the system becomes unstable, causing the network configuration to change from series to parallel. When the network is connected in parallel, the traffic load is shared among the links, causing a lower jitter value as well as only a minor percentage of packet loss. As a result, the quality of the video restores back to the original. The feedback is applied after 150 seconds. We can observe that after the application of the feedback the network does

FIGURE 5.9. Screen shot from the video taken when the network is stable

not restore back to the stable state, right away, as in the cases of marking down DSCP and dropping out-of-profile packets. Since it takes approximately 60 seconds to establish a link, the network restores back to stable state after t=200s

The following video screenshots are taken from [**38**] is used in our experiment.

5.5. Conclusion

Our feedback mechanism has a positive effect on the real-time traffic. The network goes to the stable state in real-time in all the scenarios except for the parallel path where there is a slight delay. The delay is due to the time is taken for the link to be enabled. In this approach, we did not consider the RTP delay, which is an important quality of service parameter for video streaming, as we did not observe a drastic change in the delay. We used a UDP flood attack to emulate the DDoS attack. In our approach, the micro firewall rule detects the known traffic and allows the video streaming seamlessly despite the attack. This approach is not suitable for data traffic as the rate of data is unpredictable; also, prioritizing the data traffic is not an acceptable approach. Although we did not actively stop the attack,

FIGURE 5.10. Screen shot of the video when the network is under attack. We can observe the video is completely distorted and the Quality of Experience (QoE) is very poor

FIGURE 5.11. Video screenshots taken instantly after the feedback was provided. We can observe in the first image that the video is not completely restored. In the second image, the video is completely restored and is streaming seamlessly

except for dropping the out-of-profile traffic scenario, we were able to use a passive approach to bring the network back to a stable state.

5.5.1. Future Work

In the future, we will design the controller to automate the feedback process. The controller will detect the irregularities in the output and add the corresponding rule to ensure seamless video streaming. We will test this feedback mechanism with several other types of DDoS attacks such as SYN flood and other application layer attacks, similar to the one considered in [**108**], on the Real-Time services. We would like to extend this feedback mechanism in the Software Defined Networks (SDN) architecture where control theory proves to be more effective.

CHAPTER 6

AUTOMATIC FEEDBACK CONTROL FOR GRACEFUL DEGRADATION OF REAL-TIME SERVICES IN THE FACE OF AN ATTACK

6.1. Introduction

Distributed denial of service (DDoS) attacks are one of the biggest cyber threats for many businesses such as financial sectors, IT services, and telecom services and with the growing number of Internet of Things (IoT) devices connected to the internet, this trend is expected to continue [**82**]. After the fourth quarter of 2017, DDoS attacks increased by 91% [**9**]. According to a Verisign DDoS trends report [**8**], the average attack peak size increased to 850%. According to the same report, out of the different types of DDoS, UDP floods topped the chart with 42%.

On February 28th, 2018, Github suffered from the most massive DDoS attack ever [**14**] with a jaw-dropping amount of traffic. The rate of the attack recorded was around 1.2 Tbps. Another high profile attack was on October 21, 2016, that disrupted several services on the internet. The attack rate recorded was around 600 Gbps. A few months later, Microsoft's instant messaging service, Skype, suffered from connectivity issues due to an alleged DDoS attack [**25**]. The outage lasted for days, and the users were unable to communicate with each other. These DDoS attacks make services unavailable to the users. To ensure a user's quality of experience (QoE), we need to be able to identify and mitigate these attacks. However, distinguishing between legitimate traffic and attack traffic is challenging.

6.1.1. Motivation

DDoS attacks are evolving, and hackers are unleashing new techniques to amplify the attack. The financial impact of these attacks is growing. The average cost of a DDoS attack on an enterprise is over $2M per attack and is rising dramatically [**85**]. These attacks not only impact the financial costs but also damage the reputation of the organizations. With attacks increasing rapidly, the design of resilient systems is of utmost importance. Resilience is the ability of a system to withstand when provoked by an external disruption. Disruption

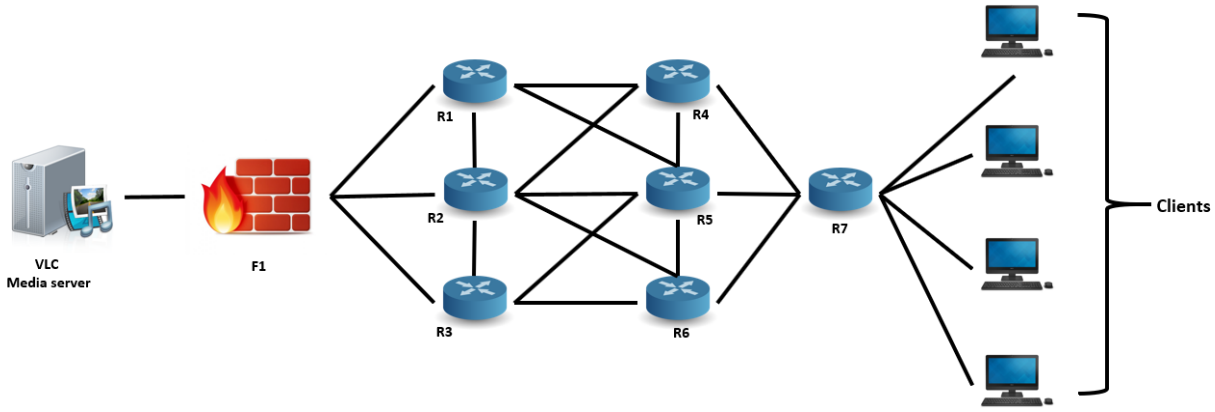

FIGURE 6.1. Experimental network topology: network elements connected in a mesh topology. All the network devices (F1: firewall, and R1-R7: routers) are configured with open shortest path first (OSPF) routing protocol to route the packets. The routers are configured with a round robin load balancing feature to share the load equally among the links.

or disturbance is an abnormal activity that hinders the normalcy of the system. Resilient networks try to provide the desired level of service, despite challenges such as malicious attacks, and misconfigurations.

In our previous work, we developed a robust closed-loop feedback mechanism to maintain the stability of a web service in the face of an attack [**108**]. We presented a passive approach of distributing the load to multiple links and reducing the impact of an attack, using feedback control. The results proved that the proposed feedback mechanism provided a positive effect on the system and the web services were restored in real time. The impact of the attack was reduced in about 60 seconds, and the user quality of experience was maintained, making the network stable.

Our work focuses on designing a robust closed-loop feedback control mechanism to protect the network and provide resilience to a network when triggered by attacks and faults. We develop this feedback control mechanism for real-time traffic, such as streaming video and audio, which is very sensitive to delay, packet loss, and jitter. Slight disruptions in the

network can hinder the performance of the real-time traffic. Network traffic such as the real-time traffic is highly dynamic, complex and very unpredictable. Understanding this complex nature of the network traffic is critical in being able to design an accurate model. In our approach, we look to system identification to build and validate a model for the complex and dynamic network.

Current mitigation techniques ranging from hours to days are entirely unacceptable given the cost and inconvenience these attacks place on our society. Our mechanism provides a real-time and scalable solution to maintain the service level agreements and to deliver video and audio streaming seamlessly smooth in spite of an attack. After designing the dynamic feedback controller, the question arises can we quantify resilience, if so, how? We look to control system theory to define the metrics of the resilient system.

6.1.2. Related Work

Feedback control theory has been applied to many computing and networking systems. However, there is not much research on the use of feedback control to detect and control the attacks. Ram Dantu et al.,[**52**] propose an Automated Defense System based on feedback control theory to control the spreading of a worm in a network. They design a state-space model that detects and controls the virus by measuring the velocity of the number of new connections. The authors design a PI controller that limits the number of connections. The authors claim that they were able to limit the infection to less than 5% of the hosts.

Feedback control theory has been applied to various areas including performance modeling of video streaming [**55**–**57**, **77**, **103**, **113**]. A Quality Adaptation Controller for live adaptive video streaming based on feedback control theory was presented by Luca De Cicco et al., [**58**]. This controller throttles the video level. The controller tracks the queue length and outputs the preferred bitrate of the video to match the available bandwidth. The authors claim that they achieve this in less than 30 seconds.

Guibin Tian and Yong Liu [**103**] designed and implemented a receiver-driven adaptation algorithm for Dynamic Adaptive Streaming over HTTP (DASH). They design a PID controller driven by deviation in the client-side buffered video time. The authors claim that

their algorithm provides a balance between the needs of video rate smoothness and high bandwidth utilization.

The approaches as mentioned earlier focus on performance and regulating the bitrate of video streaming. To the best of our knowledge, the use of feedback control to minimize the impact of an attack has to be yet explored especially for real-time traffic. Our approach decreases the degradation of real-time services. The feedback mechanism implemented in our approach minimizes the impact of the attack and restores back the QoE of the network service.

6.2. Architecture

6.2.1. Network Topology

The experimental testbed implemented in our lab is shown in Fig 6.1 in our lab. The network comprises a firewall, routing devices, media streaming server, and clients. We implement a mesh network using the Cisco Catalyst devices and the firewall. These devices are configured to use Open Shortest Path First (OSPF) protocol to route the packets. We also set the routers with a load-balancing feature such that the load is shared equally among the links.

In this architecture, a centralized controller collects information such as arrival rate, inter-arrival time, jitter, packet loss and various other metrics. This controller is aware of the network topology, configurations of multiple network devices, security policies and the service level agreements. This controller is a logical function and can be implemented in any network device such as routers, load-balancers, and also in any security devices such as firewalls and intrusion detection systems (IDS). In our proposed approach we implement the closed loop feedback mechanism in the firewall which is capable of controlling the micro-firewall rules to stabilize the network under attack. The measurements such as packet loss and, jitter are fed back into the firewall to understand the current state of the network.

6.2.2. Client

We use the VideoLAN client (VLC) media player [**34**] to play the streaming video and audio content. Multiple clients installed with VLC media players were used to playing the same content from the streaming server.

6.2.3. Server

A standalone media server is used to host all the media content and to stream to all the devices over the network. This server is equipped with an Intel Xenon processor (4 cores) and a 32 GB RAM. We consider three types of video qualities a) 2K Video (High quality), b) 720p (High Definition), and 480p (Low quality).

6.3. Methodology

6.3.1. System Identification

To be able to design a robust feedback control architecture and to identify the characteristics of the complex network a good system model is required. Due to various working conditions of the network, building an accurate model to capture the entire dynamics of the network is challenging. However, based on the observations of the network in certain conditions, we can dynamically predict a model. We are thus motivated to look into system identification technique to determine the dynamics of the complex network comprising of clients, servers, routers, firewalls, and so forth.

System identification is a process of deriving mathematical relations between input and output data [**80**]. We use a black box approach [**79**] [**2**] to identify the model. We consider the network comprising of routers, firewall, clients and media server as a plant. We model the plant using the system identification technique. The inputs considered are the bitrate of the video and the committed information rate (CIR). The output measured from the plant are the QoS metrics such as packet loss, jitter, and bit-rate of the video. Control input is a parameter which can be dynamically adjusted and can affect the behavior of the system. Hence, we chose the Committed information rate (CIR) as the control input. Details of the selection of control input are discussed in section III B.

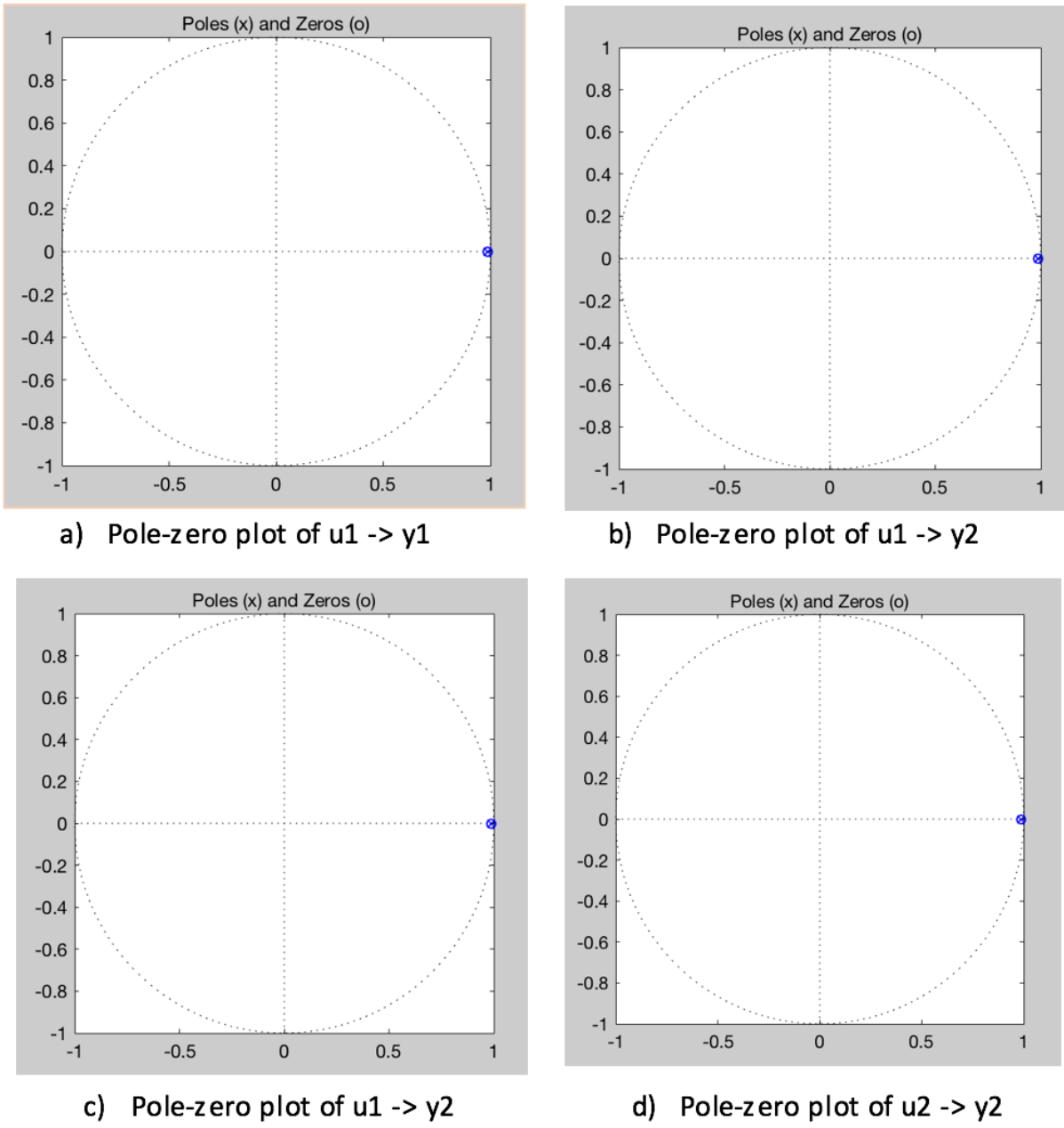

FIGURE 6.2. Pole-zero plots of the identified model. According to control theory [**71**], location of poles and zeros define the stability of the system. Here, the poles and zeros lying within the unit circle indicate the model is stable.

State space models have provided better representations of various classes of engineering, computing and several biological systems and processes [**71**]. State space models are used to characterize the system, i.e., how the system functions or performs based on the

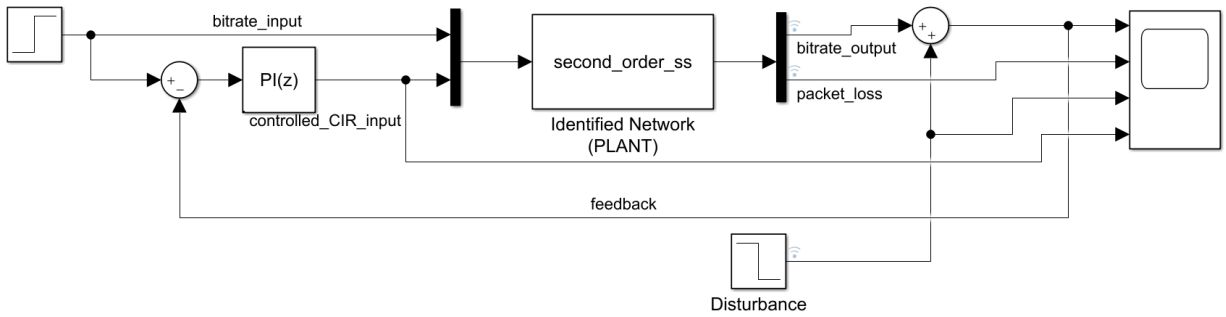

FIGURE 6.3. Simulink model of our closed-loop feedback control architecture. The plant is the entire network comprising of all the devices including firewall, routers, clients, and servers. The output from the plant, i.e., the QoS metrics such as bitrate and packet loss are fed back into the PI controller. The PI controller designed is the brain of this system, that regulates the process. When the network is disturbed by an attack, the controller provides a controlled input, i.e., CIR, to the plant. Bitrate and packet loss are provided as a reference input to the controller. The disturbance is a negative step input added to the bitrate.

state variables which explain the dynamics of the system [**71**]. State space models are also scalable and can be applied to non-linear systems. Hence, we decided to use a state space model for modeling our network.

The general format of a discrete-time state space model is:

$$x(t+Ts) = Ax(t) + Bu(t) + Ke(t)$$
$$y(t) = Cx(t) + Du(t) + e(t) \tag{3}$$

where u is the input, x is the state, y is the output and e is the error. A, B, C, D, and K are matrix coefficients and must have these characteristics:

A must be an n-by-n matrix, where n is the number of states.

B must be an n-by-m matrix, where m is the number of inputs.

C must be an r-by-n matrix, where r is the number of outputs.

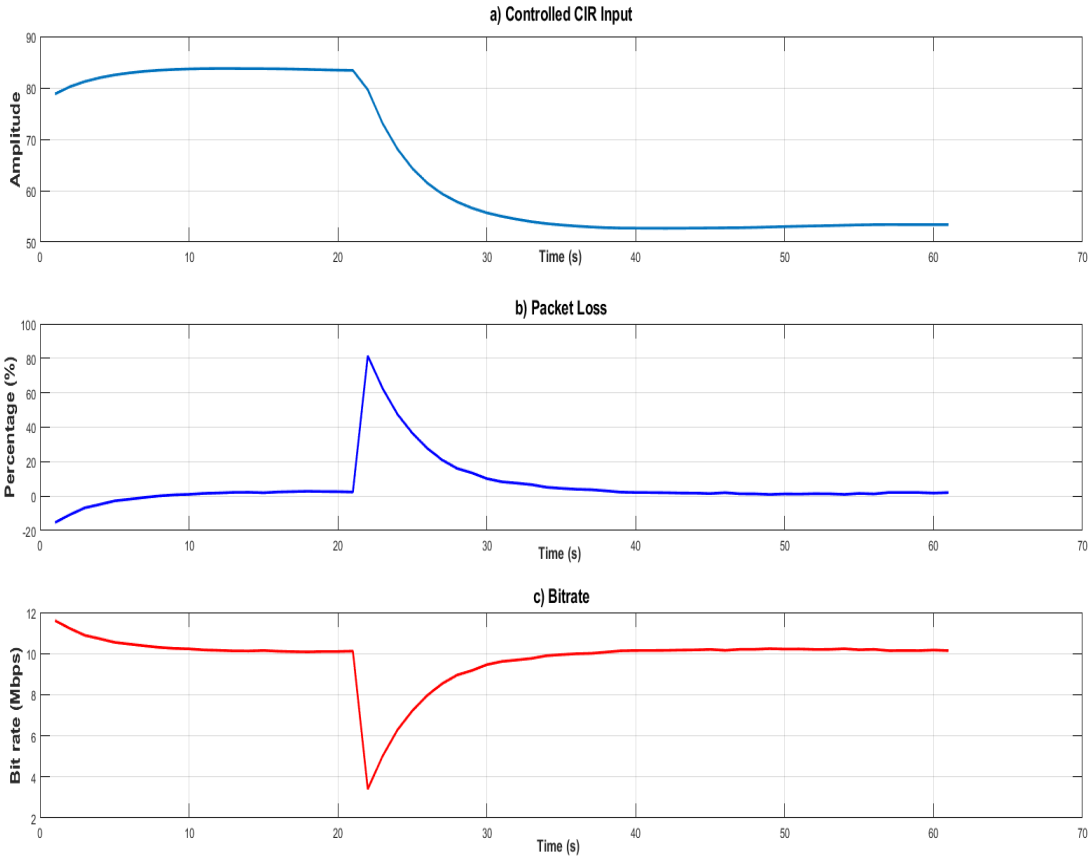

FIGURE 6.4. Results from the simulation of the PI controller to control the impact of an attack and maintain the QoS. We can observe that when the system is excited with an attack, the PI controller regulates the CIR value, ensuring the packet loss and bitrate are maintained at the desired range, in just about 15 seconds.

D must be an r-by-m matrix.

We use the the system identification toolbox of matlab [**28**] to construct the state-space model for the network system. The following are the parameters estimated using the toolbox:

$$A = \begin{bmatrix} 0.9895 & 0.000192 \\ -0.002089 & 0.9885 \end{bmatrix}$$

$$B = \begin{bmatrix} 2.266e-06 & -3.122e-07 \\ 1.347e-05 & -1.857e-06 \end{bmatrix}$$

$$C = \begin{bmatrix} 8.822 & 1.949 \\ -102.6 & -31.67 \end{bmatrix}$$

$$D = \begin{bmatrix} 2.438 & -0.2546 \\ -19.38 & 3 \end{bmatrix}$$

$$K = \begin{bmatrix} 0.4056 & 0.02522 \\ -1.341 & -0.1155 \end{bmatrix}$$

According to control theory [**71**], the stability of a model is defined by the location of the poles and zeros. In a transfer function, poles and zeros are the roots of the numerator and denominator polynomials respectively. The pole-zero plot of the identified model is shown in Fig 6.2. We can observe from the figure that the poles and zeros lie within the unit circle indicating the model is stable [**71**].

6.3.2. Feedback Control

Our goal is to make the network stable and resilient by maintaining the QoS and the SLA of the network service, in spite of an attack. During an attack, the network is congested, due to which the QoS of the real-time traffic drastically drops as shown in Fig 6.5 and Fig 6.7. From the figures, we can observe that as the rate of attack increases the quality of both audio and video degrades progressively. In order to achieve our goal, the attack must be managed. We propose a robust closed-loop feedback control approach to limit the degradation of the real-time traffic gracefully. While designing a controller, we address two critical questions:

(1) What can be done to control the traffic rate

(2) When is the right time to introduce feedback

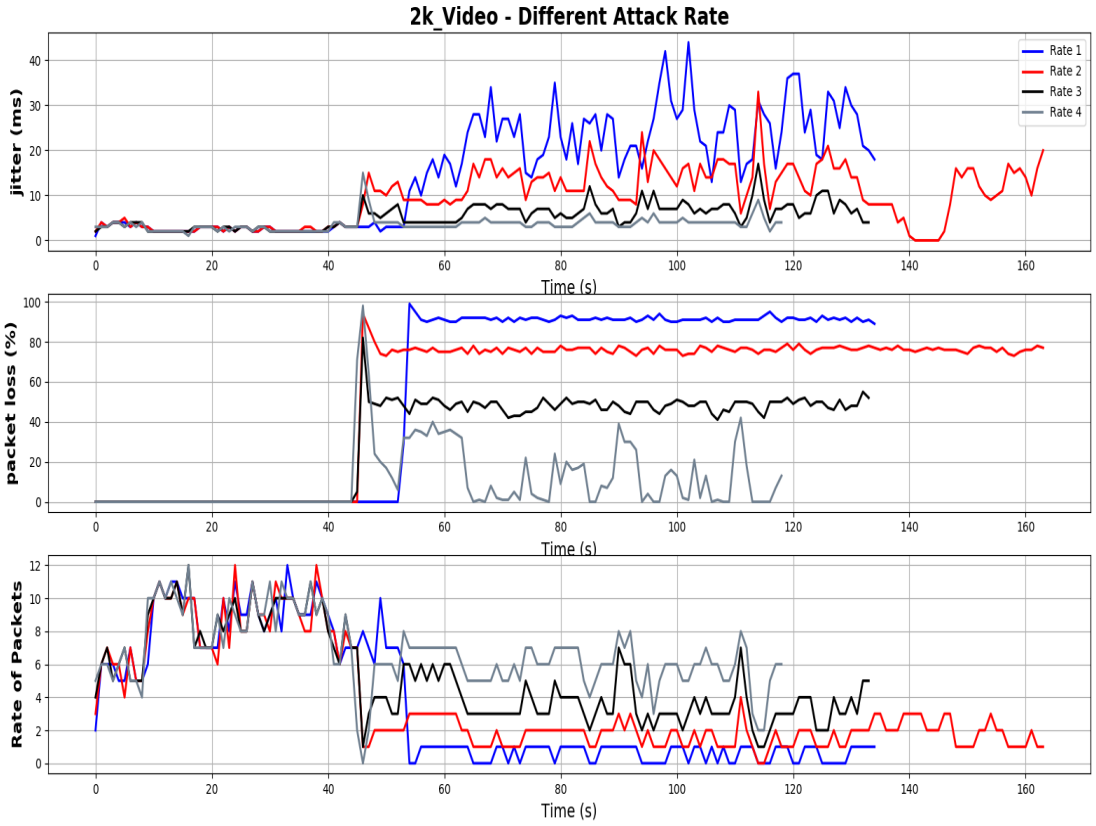

FIGURE 6.5. (a) Jitter, (b) packet loss and (c) bitrate measured from the network before and after the attack while streaming video. Under normal conditions, i.e., from t = 1 to t = 40 seconds, the network is stable, and the video streaming is seamlessly smooth. After 40 seconds, we flood the network with different rates of attack as shown in Fig 6.6. We can observe the degradation of the video traffic against various rates of attack. As the rate of attack increases the QoS of video traffic drops drastically. All the attacks deter the video streaming.

The answer to the first question lies in regulating the attack traffic by using a micro-firewall rule known as committed information rate (CIR).

Drop Out-of-Profile Traffic

This rule drops out-of-profile traffic based on the committed information rate (CIR) and burst size parameters. The CIR defines the number of tokens removed at each interval, and the burst determines the maximum amount of packets (in megabytes) the bucket can hold at any time. The CIR parameter policies or drops the excess traffic that does not comply with the policy, i.e., if the traffic flow reaches the configured CIR rate, the excess traffic is dropped. We consider the CIR as a control input which regulates the flow of traffic. We now understand that by tuning the CIR value we can control the traffic rate. However, the most important questions arise, i.e., question two, when is the right time to introduce feedback and what can be done to control the CIR parameter dynamically. The answer to this question lies in designing a suitable controller that offers to regulate the attack traffic.

Controller Design

A proportional-integral-derivative controller (PID controller) is a closed-loop control mechanism often used in many industrial chemical and computing systems. In a closed loop control system the current output measured from the system, also known as a process variable, is fed back to the controller along with the desired reference value. The reference value is also called as a setpoint. The controller continuously calculates the error e(t) measured from the process variable and the setpoint and applies a correction to the system based on the P, I and D terms. These three variables determine the controller's behavior. The general equation of a PID controller is given below:

$$u(t) = K_p e(t) + K_i \int_0^{t'} e(t')dt' + K_d \frac{de(t)}{dt} \tag{4}$$

The proportional (K_p) variable adjusts the system output proportionally to the error signal by controlling the proportional gain of the controller. Isolated use of a proportional controller might help in reducing the error, but the final output might result in oscillations such as an

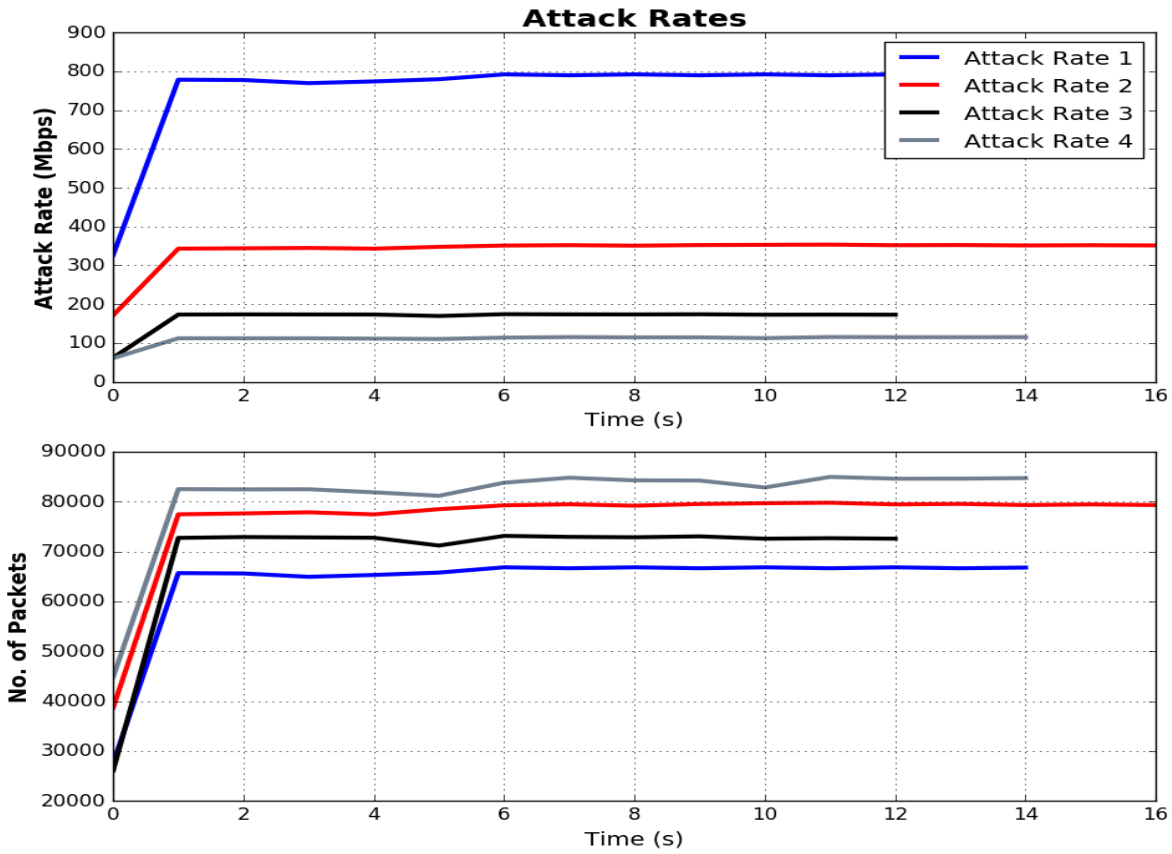

FIGURE 6.6. Attack rates used to emulate the real world DDoS attack (a) rate of packets or bitrate in Mbps, and (b) No. of packets. The attack rates are varied by varying the packet size.

on-off signal. These oscillations can be reduced by the integral variable (K_i). The derivative term (K_d) estimates the future trend of the error based on the current rate of change of the output. However, after applying the integral term the error signal converged with the output, indicating a steady state, which led us to chose a PI controller.

The PI controller implemented is shown in Fig 6.3. This controller controls the attack traffic by regulating the CIR parameter. The QoS metrics measured are fed back into the PI controller, which uses this feedback to detect anomalies. When the network is under attack, the QoS of the network service drops drastically, and the network becomes unstable. This change in the QoS values such as bitrate, packet loss, and jitter is detected by the controller,

which determines an appropriate CIR value to bring back the network to the preferred QoS values. During simulation, we use a negative step input as an output disturbance to the plant which reduces the QoS values of the real-time traffic.

Tuning a PI controller means selecting the best values for the P and I variables to achieve the desired results. In our experiment, we want to contain the attack traffic when an attack is detected. Fig 6.4 shows the results of a simulation of applying a PI controller to the network designed in Matlab. We can observe that during an attack the bitrate of the video drops to less than 1 Mbps. When the feedback is provided using P = 0.047 and I = 0.094 the system converges to the setpoint value with an initial overshoot.

6.3.3. Generation of Traffic

We use VLC media player to broadcast a stream. We select the Real Time Streaming Protocol (RTSP) as the streaming method. We consider three types of video qualities a) 2K (High quality) b) 720p (high definition) and c) 480p (standard definition) and a high-resolution audio (320 Kbps). The chosen video was streamed across the network shown in Fig 6.1. We use the tcpdump packet analyzer to sniff the packets at the server and client's interface, which acts as a sensor. The data such as rate of packets, inter-arrival time, etc. is collected periodically from the packet capture. We use this collected data as input into our model.

6.4. Analysis of Results

We study the performance of the feedback controller architecture by emulating a real-world network. The firewall and the routers are connected in a mesh topology as shown in Fig 6.1 and configured with OSPF protocol. We attempt to model the steady state of the network from the input-output data collected when the network is stable, i.e., when the network is under normal conditions, and any attacks are not disrupting the network services.

We then flood the network with a large number of UDP packets. The attack rate is varied by varying the packet size. Fig 6.6 shows the different rates of attack used to stress the network. We model these attack rates that resemble a step input as a disturbance input

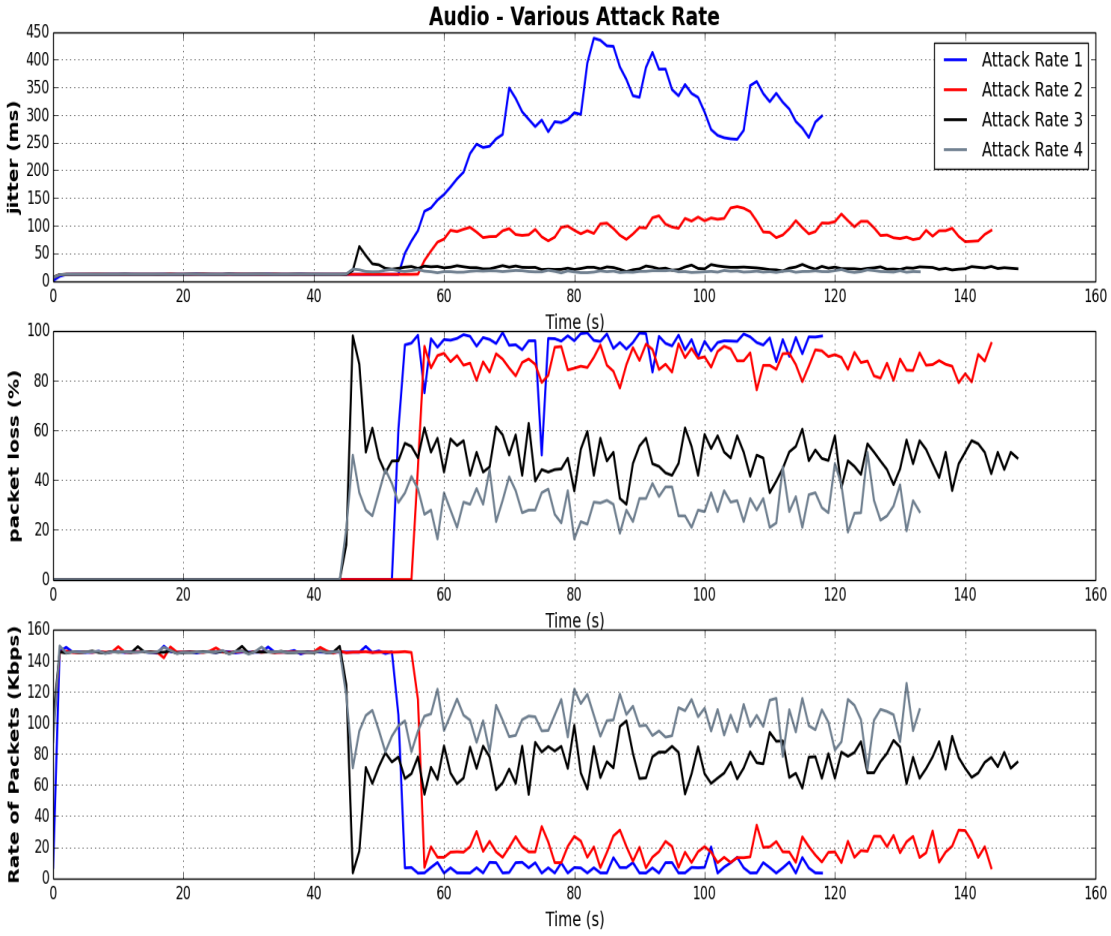

FIGURE 6.7. (a) Jitter, (b) packet loss and (c) bitrate measured from the network before and after the attack while streaming audio. Under normal conditions, i.e., from t = 1 to t = 40 seconds, the audio is seamlessly smooth. After 40 seconds, we flood the network with different rates of attack as shown in Fig 6.6. We can observe the degradation of the audio towards various rates of attack. During a very high rate attack, the bitrate of the audio drops down to 1-2 Kbps. The packet loss percentage increases to 98% leading to a very high jitter value of 400 ms. We can observe that as the rate of attack decreases, the QoS metrics, i.e., packet loss percentage and jitter decreases but unacceptable.

to the plant. When the network is under attack, the network service is hindered resulting in the degradation of the QoS. Fig 6.5 and Fig 6.7 show the effect of these attacks on video and audio traffic simultaneously. In spite of both video and audio traffic being prioritized, the attacks impact the legitimate traffic. We can observe from the Fig 6.5 that when the attack rate is very high, the bitrate of the video reduces resulting in more than 95% of packet loss increasing the jitter. Similarly as shown in Fig 6.7 the audio traffic can also be impacted by the attack increasing the jitter to nearly 400ms which is unacceptable.

6.4.1. Applying Feedback

To preserve the service level agreements of the network service the feedback to the network is applied by adding a micro-firewall rule that controls the rate of traffic during an attack. Two different types of network services are considered to investigate the behavior of the feedback controller: 1) Video and 2) Audio. We also test the controller performance with various video and audio resolutions. We consider bit-rate of the video as a reference input to the plant. Bit-rate of real-time traffic such as audio and video corresponds to the quality of the audio/video traffic, higher the bit-rate, better is the quality of the real-time traffic.

We conducted experiments by providing two different reference inputs to the controller 1) bitrate and 2) Packet Loss. We present a detail discussion about the two scenarios in the next sections.

6.4.2. Bitrate as Setpoint

In this section, we explain the performance of the PI controller when the bitrate is set as a reference input to the controller. Fig 6.8 shows the QoS metrics (output) of video traffic collected from the network before and after introducing the feedback to the network. After 25 seconds we flood the network with a large number of packets, to emulate a DDoS attack. When the network is under attack, the nodes are congested resulting in the dropping of packets. During the attack, we can observe from the Fig 6.8 (b) that the measured percentage of packet loss increased to 80%. Due to a very high drop in the packets,

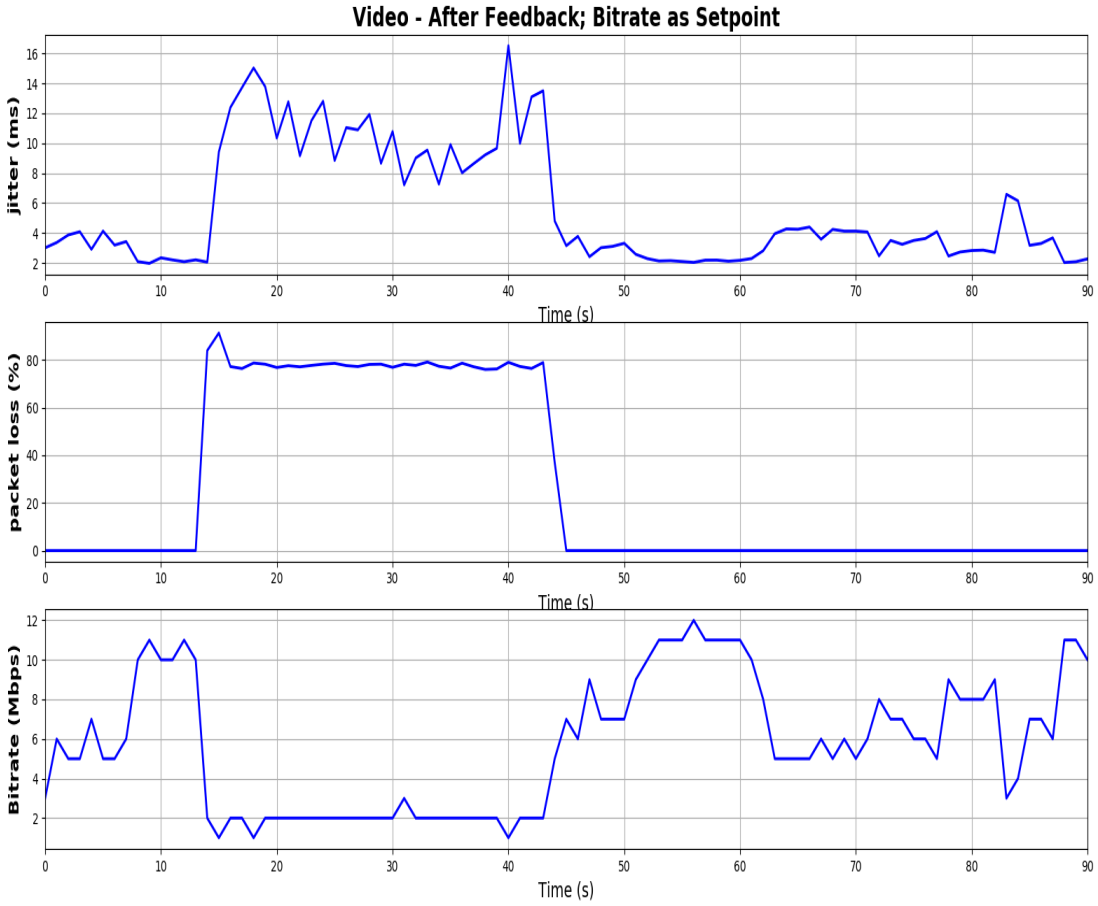

FIGURE 6.8. (a) Jitter, (b) packet loss and (c) bitrate measured from the network before and after providing feedback. The network is attacked after 15 seconds. We can observe, when the network is under attack, the percentage of packet loss increased to 80%. Due to this high packet loss, the jitter increased by nearly 83%. At t = 40s we apply feedback to the network under attack. After the feedback is applied, the service returns to normalcy, maintaining the QoE and stabilizing the network under attack. The reference input considered in this scenario is Bitrate.

(c) that the bitrate of the video dropped to nearly 2 Mbps from 11 Mbps.

We apply the feedback to the controller after 40 seconds. We can observe that, when feedback is provided to the network after 40 seconds, the network returns to the stable state in less than 10 seconds. After implementing the feedback, the PI controller detects the drop in the bitrate of the video traffic. The controller then tunes the CIR parameter to a lower value to ensure the network service is maintained at the desired bitrate. After the controller updates the micro firewall rule into the firewall, we can observe the attack is contained and the video is restored back in real time.

6.4.3. Packet Loss as Setpoint

In this section, we explain the performance of the PI controller when the packet loss is set as a reference input to the controller. Fig 6.9 shows the output collected from the network excited with a different rate of attack. In this experiment, we flood the network with a low rate attack after 25 seconds. We then provide the feedback after 55 seconds. The controller detects the drop in the QoS values and controls the non-prioritized traffic by tuning the CIR to a lower value. In this experiment, the reference input to the controller was set to maintain the packet loss below 2%. Hence, from the Fig 6.9 (b) we can observe a very few percentages of packets are lost after providing the feedback.

6.5. Conclusion

The proposed feedback mechanism provides a real-time and scalable solution to make the network persistent during an attack and delivers video streaming without any degradation during an attack. In the conducted experiments the PI controller maintained the network service in a stable state in spite of an attack in less than 20 seconds. The closed-loop feedback mechanism designed proved that in every attack scenario, the service level agreements of the real-time services are not violated. The PI controller is designed as a disturbance rejection controller. The PI controller designed is a multiple loop SISO controller. We look to improve the controller and make the system more robust by developing a MIMO controller.

In our approach, the micro-firewall rule detects the prioritized real-time traffic. This

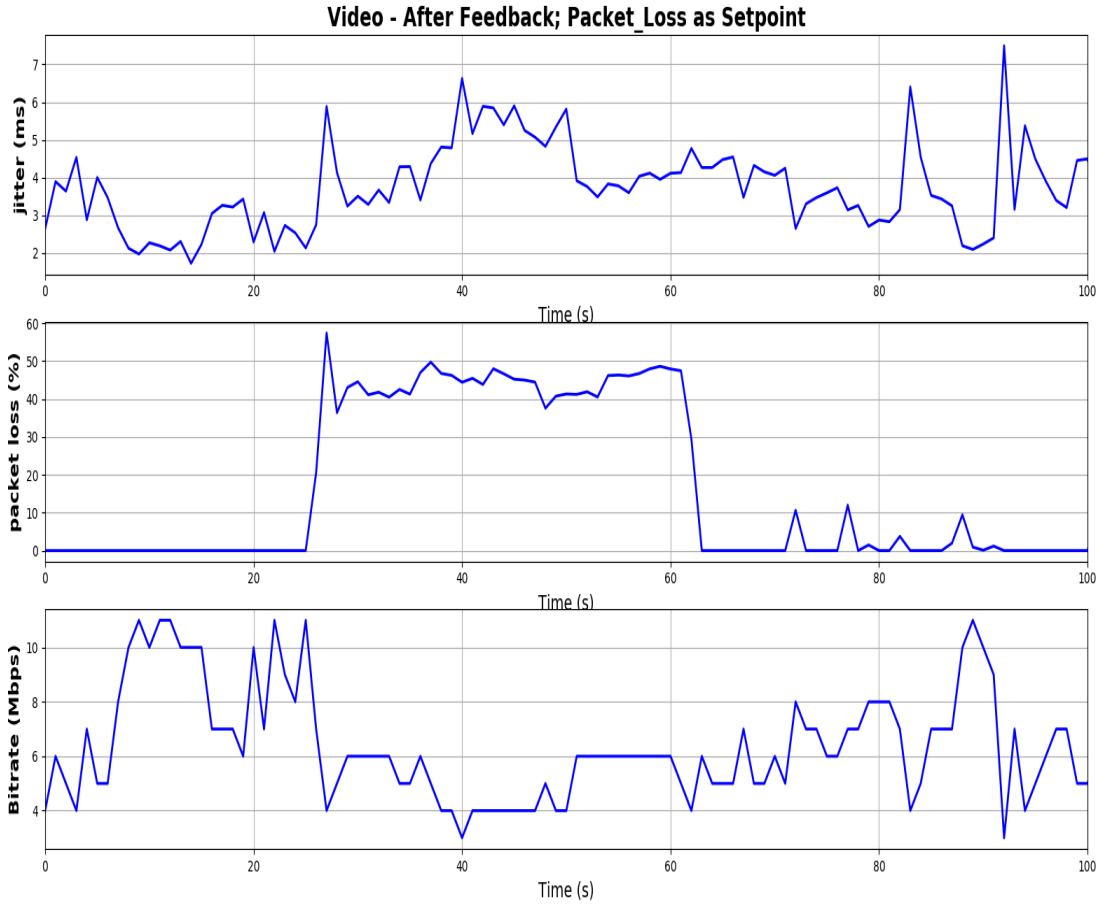

FIGURE 6.9. (a) Jitter, (b) packet loss and (c) bitrate measured from the network before and after providing feedback. The network is attacked after 20 seconds. We can observe, when the network is under attack, the percentage of packet loss increased to 50%. Unlike in the previous scenario Fig 6.8, the QoS of the real-time traffic dropped by nearly 50% which is unacceptable. The feedback is applied after 60 seconds to the network under attack. In this scenario, packet loss is set as a reference input. After the feedback is applied, the service is restored.

approach is not suitable for an attack on data traffic, as the mechanism is only capable of identifying prioritized traffic and prioritizing non-real-time traffic is not an acceptable method. In our approach, we did not consider the end-to-end delay and latency of the real-time traffic, which are an essential quality of service metrics because we observed a lot of artifacts due to high packet loss. Also, we only considered UDP flooding as a significant source for the DDoS attack. This approach also holds good for TCP based DDoS attacks such as SYN flooding, where the controller would tune the burst size parameter to minimize the number of TCP connections.

6.5.1. Future Work

In the future, we will fine-tune the model to reduce the overshoot of the controller and also look to design more stable controller. We will also test the controller's performance with various other types of attacks such as worm propagation, and SYN flooding. We also look to design a predictive controller such as a model predictive controller (MPC) which provides the intelligence of predicting the attack. We would like to extend this closed-loop feedback mechanism in Software Defined Networks.

CHAPTER 7

DYNAMIC AND PREDICTIVE CONTROL FOR GRACEFUL DEGRADATION OF REAL-TIME SERVICES IN SDN ENVIRONMENT

7.1. Introduction

Distributed denial of service (DDoS) attacks continue to plague businesses and consumers alike, and due to the evolving nature of cyber-physical systems, these attacks are expected to grow in volume and complexity. Despite a number of security tools, the attacks are still encountered and are causing various businesses to sweat. After the fourth quarter of 2017, DDoS attacks increased by 91% [**9**]. According to a Verisign DDoS trends report [**8**], the average attack peak size increased to 850%. According to the same report, out of the different types of DDoS, UDP floods topped the chart with 42%.

The GitHub attack previously mentioned in Chapter 3 serves as a stark reminder of the perils of unregulated network traffic. Another high profile attack was on October 21, 2016, that disrupted several services on the internet. This massive attack took down a significant portion of the internet services on the U.S. east coast and in central Texas. As a result, various internet services such as Twitter, Reddit, Amazon, Spotify were not available to the users. The rate of the attack recorded was around 600 Gbps. A few months later, Microsoft's instant messaging service, Skype, suffered from connectivity issues due to an alleged DDoS attack [**25**]. This outage lasted for days, and the users were unable to communicate with each other. The immediate fallout of these attacks is apparent in lost revenue and loss of consumer loyalty, which are crucial to network service providers. To ensure a user's quality of experience (QoE), we need to be able to identify and mitigate these attacks. However, distinguishing between legitimate traffic and attack traffic is quite challenging.

With attacks growing in number, the design of a resilient network is critical. Resilience is the ability of a system to withstand when provoked by an attack. Resilient networks have the ability to absorb the disturbance and provide the desired level of service. A desirable

attribute of resilient networks is the ability to configure the network dynamically. The emerging paradigm of Software Defined Networking (SDN) that attracted attention in recent years promises to deliver the elasticity in configuring various network devices, such as routers, switches, firewalls and other security devices.

With the deployment of SDN, the network performance can be improved by providing effective traffic engineering. Microsoft designed a software-driven WAN which interconnects the data centers and achieves high network throughput [**73**]. The flexibility of SDN architecture has convinced Google to adopt it in its data centers [**74**]. The SDN architecture also finds applications in various types of networks such as wireless, home, cellular and enterprise.

Software Defined Networking (SDN) is changing the design and management of the internet. The word SDN was coined by Kate Greene [**31**] about the OpenFlow project at Stanford. SDN is a networking architecture that decouples the control plane and the data plane [**26**]. The controller (known as the brain) is a logically centralized component [**26**] of the control plane. The controller is used for network intelligence and policymaking. It also determines the forwarding path. The data plane consists of forwarding elements and host machines. SDN radicalized the functionality of the networking area. Early SDN models focused on moving the entire control plane into the controller leaving the switching elements simple in the data plane.

7.2. Motivation

DDoS attacks are evolving, and hackers are unleashing new techniques to amplify them. The financial impact of these attacks is growing rapidly for many businesses. The average cost of a DDoS attack on an enterprise is over $2M per attack and is rising dramatically [**10**]. In addition, these attacks also damage the reputation of the organizations. Current mitigation techniques ranging from hours to days are completely unacceptable given the cost and inconvenience these attacks place on our society. Despite the existence of many cybersecurity tools and techniques, networks are still vulnerable to malicious attacks.

The main problem with a DDoS attack is that it may be difficult to distinguish between legitimate traffic and attack traffic. During an attack, the network services are

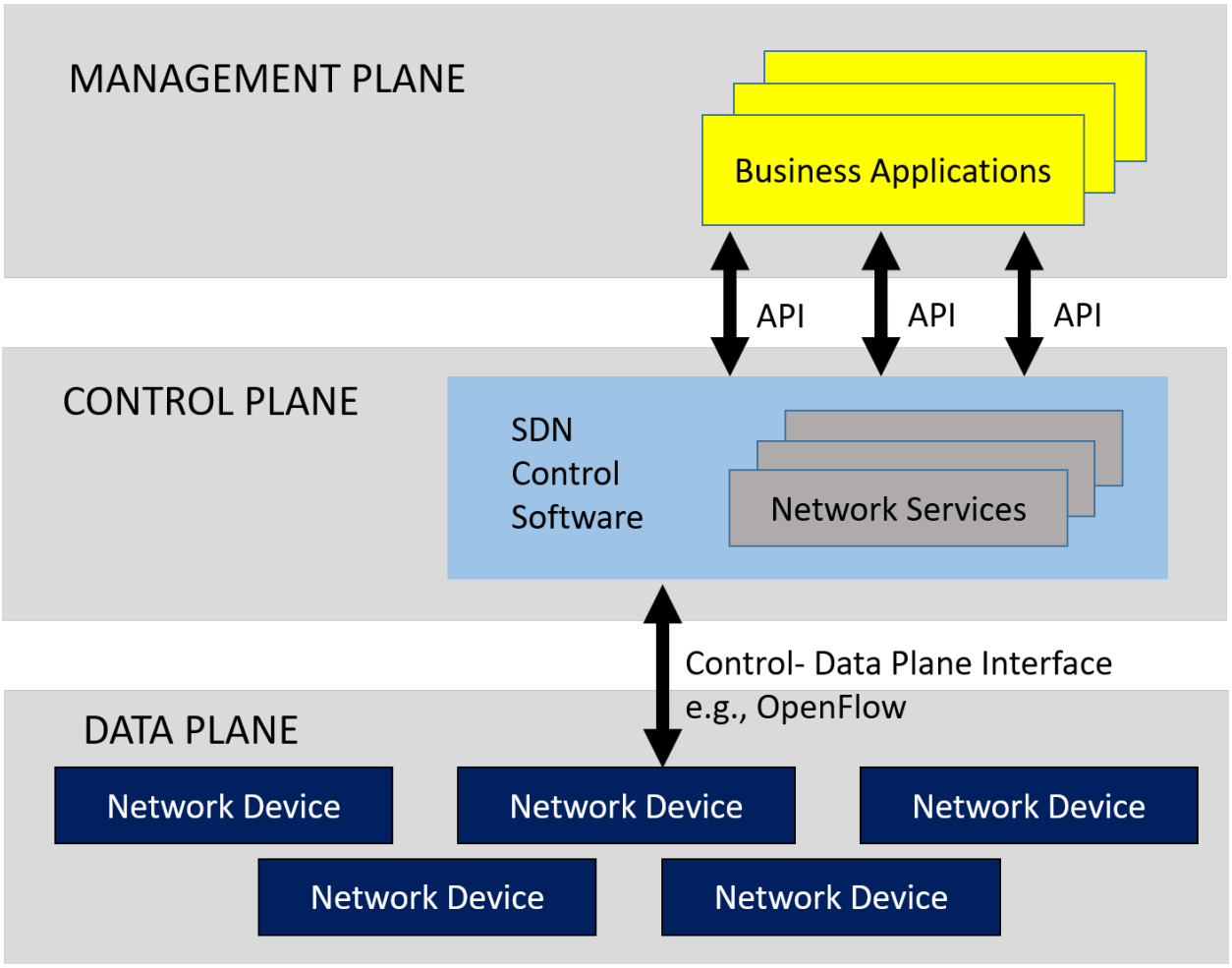

FIGURE 7.1. SDN architecture [**26**]. Business applications include network management, QoS enforcement, and load balancing. The API used to communicate with the controller is the northbound interface. The controller services include core services such as firewall, flow-entry pusher, forwarding engine. The controller communicates with the data plane through a southbound interface, OpenFlow

impacted, and the service availability drops drastically. Since networks are complex and highly dynamic, static techniques such as rate limiting, using Access Control Lists [**91**] are inadequate to respond to such types of attacks. In this regard, feedback control mechanisms play a crucial role in mitigating the attacks in real time [**52**]. Feedback control is about regulating the system characteristics with disturbance rejection.

In this chapter, the closed-loop feedback mechanism is applied to real-time services. The purpose of the controllers is to limit the impact of an attack, such that the QoS metrics of the network is consistent with those specified in the service level agreements. Service level agreement is a commitment guaranteed by the internet service providers to the user. These agreements include one or more service level objectives (SLOs), which is a criterion used to evaluate the performance of the service. These objectives include metrics such as availability, delivery time, response time, failure rate, delay, jitter and various other scalability metrics. Enforcing these objectives requires the service providers to meet their requirements by efficiently utilizing sufficient resources. Hence, implementing SLO becomes a control problem.

I put forth two novel feedback control mechanisms to minimize the effect of volumetric attacks such as DDoS. During an attack, the closed-loop feedback control mechanism detects and limits the impact of an attack on real-time services by maintaining the service level agreements (SLA). The controller makes intelligent decisions to ensure the service level objectives are met by dynamically regulating the configuration settings of the network devices. The service level objectives are provided as a reference input. The first controller is a multi-loop proportional, integral (PI) controller. The second implemented controller is a model predictive controller (MPC) which is an exceptional feedback control system that uses a model to predict the future outputs of a process. The proposed architectures are verified in a lab setup. Results from the setup show that both the feedback control models provide graceful degradation of the real-time services and the stabilize the network in real-time. We implement these feedback control strategies in an SDN platform.

7.2.1. Related Work

Control theory approach has been applied to a wide range of computing systems [7]. Xiaoqi Yin et al. [**114**] propose a control theoretic approach for dynamic adaptive video streaming over HTTP (DASH) to ensure good quality of experience. The authors implement a model predictive control algorithm for bitrate adaptation. In addition, the authors also develop a robust MPC that handles errors in throughput predictions. The basic MPC algorithm implemented by the authors has three main steps 1) predict the throughput by looking N steps ahead (which is chosen as the moving horizon), 2) optimizing quality of experience (QoE) maximization problem, and 3) applying the methodology where the player downloads the chunks. In this approach, the bitrate of the video and buffer occupancy is applied as feedback. The authors implement their algorithms in the dash.js framework and evaluate the performance of their approach with the current existing rate and buffer based techniques. The authors claim that their proposed approach outperforms existing algorithms.

A client-side controller for DASH was implemented Luca De Cicco et al. Their implementation differs from the conventional approach, which employs controllers to throttle the video level and the regulate the buffer. The authors use a single controller that throttles the video level to drive the buffer length. The control input is the buffer length, and the measured output is the video level. The authors use a feedback linearization technique to compute the control law. From various results, the authors show that their methodology performs flawless and provides high channel utilization even when the video flows share a bottleneck in the presence of TCP flows.

To the best of our knowledge, the use of feedback control to build resilient networks is yet to be explored.

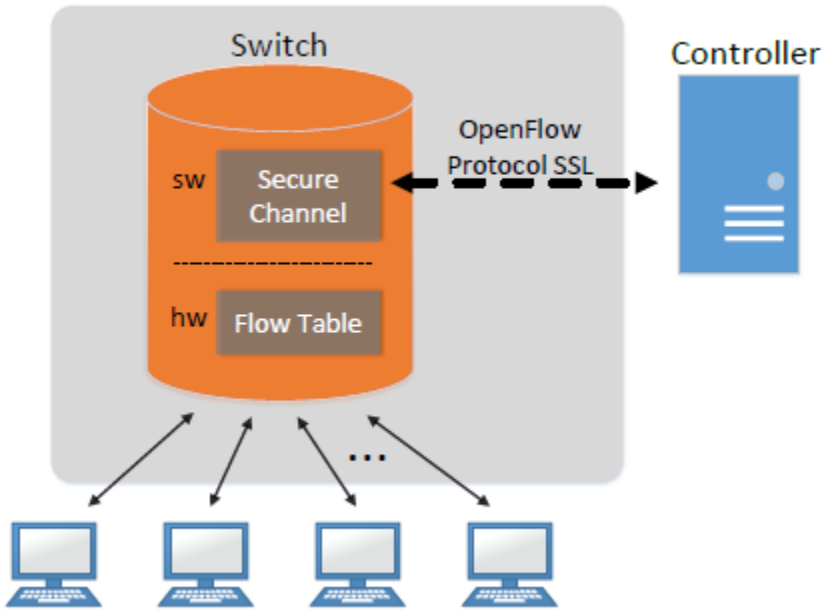

FIGURE 7.2. The basic architecture of openflow [**21**]. The controller communicates to the switches in the data plane using the openflow channel. The openflow controller installs flows in the flow table of the switch.

7.3. System Overview

7.3.1. Software Defined Network

Software-defined network is an emerging architecture that decouples the network control and the forwarding functionalities [**26**]. This architecture facilitates network management by using dynamic programs. In this architecture, a centralized controller that maintains a global view of the network controls several network devices in the data plane. The controller is managed by applications present in the management plane. The network can be configured and administered via dynamic and automated SDN programs. The SDN architecture is shown in Figure 7.1 [**26**].

7.3.2. Components of SDN

Following are the architectural components of SDN

- **SDN Applications:** SDN Applications are the programs that are designed to describe the required resources and behaviors and to perform a task in the SDN

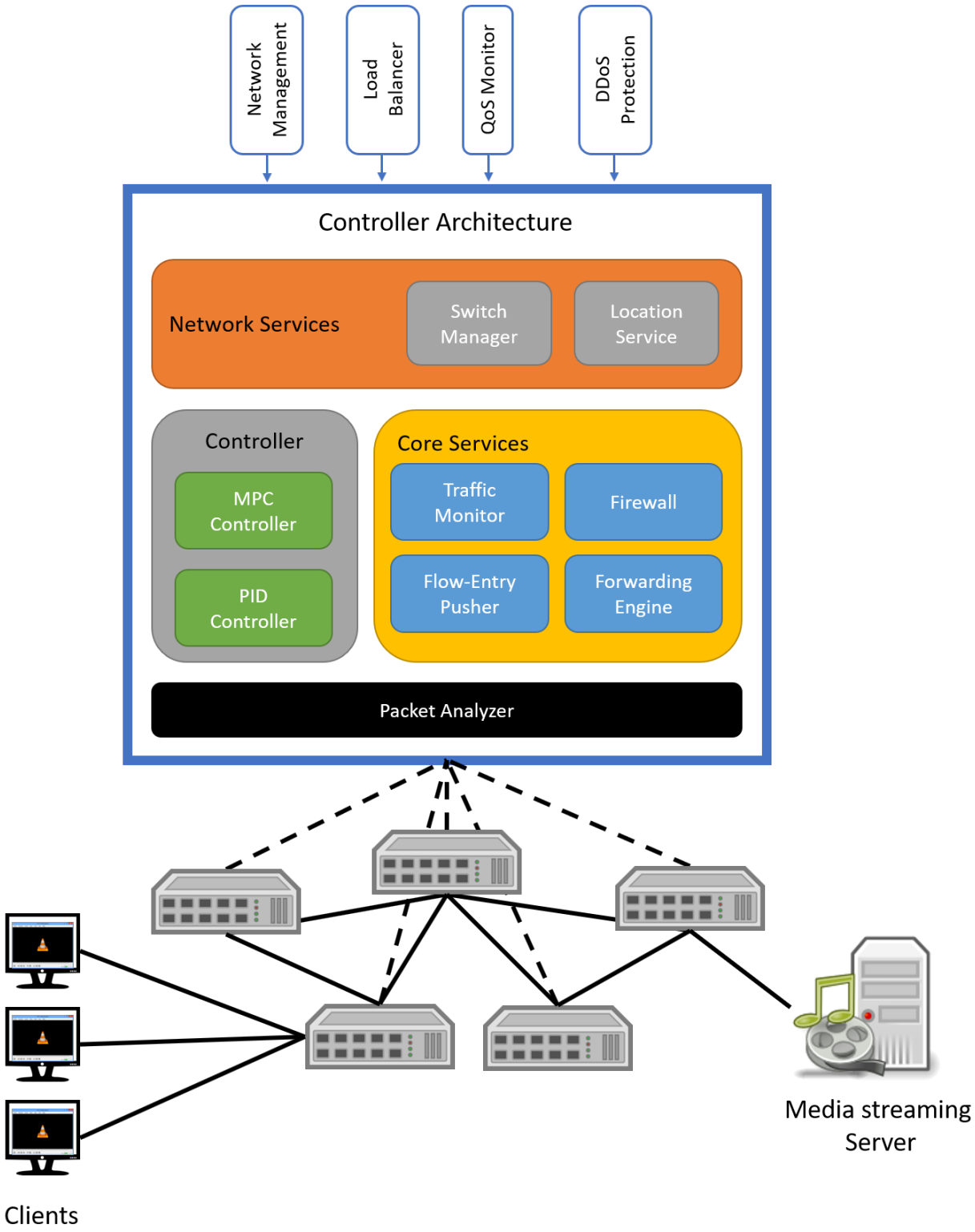

FIGURE 7.3. Software defined network topology implemented

environment. These applications use application programming interfaces (APIs) to communicate with the controller. In addition, the applications also collect information from the controller to make critical decisions such as routing, forwarding, and load-balancing. These applications include network management, load balancers, and various DDoS protection tools.

- **SDN Controller:** The SDN Controller (SDNC) is the most critical component in the SDN architecture. It is a logical entity that receives instructions from the Application layer and downloads them to the networking components present in the data plane. The controller also collects information about the network from various devices in the data plane and relays it back to the application layer. The controller achieves this using an OpenFlow channel. Every network device in the data plane is connected to the controller.
- **SDN Networking Devices:** The SDN networking devices receive instructions from the SDN controller to perform forwarding and processing of the data packets. These devices are located in the data plane, and download flows from the controller to forward packets.

7.3.3. Openflow

The controller communicates with the devices in the data plane using OpenFlow [**21**]. OpenFlow is a standard communication protocol for controlling the network within the SDN architecture. It is an open standard managed by Open Networking Foundation [6]. It is the first standard communication interface defined between the control and forwarding layers of an SDN architecture [7]. When a flow is initiated in the network, the first packet is pushed into the controller by the OpenFlow-enabled switch. The controller computes the forwarding path and sends the packet out to the switch in the data plane along with the flow modification entry. All the succeeding packets described by the flow are forwarded in the data plane. The basic architecture of OpenFlow is shown in Figure 7.2

7.3.4. QoS and SDN

QoS (Quality of Service) is a measurement of the overall performance of a network service. It is a technology used to classify data based on the priority and transfer it. It can also be used to reserve bandwidth for particular traffic such as real-time traffic. Due to new protocols such as OpenFlow, provisioning of network services is implemented through various mechanisms in SDN. SDN provides many QoS operations. The most commonly presented and used in this chapter are Differentiated services (DiffServ) and Meter Table.

7.3.4.1. Differentiated Services

Differentiated service is an architecture that is used to classify and manage network traffic to provide a better quality of service. It achieves this by assigning Differentiated services code point (DSCP) values to the packets. DSCP is a 6-bit value present in the IP header of a packet. The values in the field determine the type of service. The network devices scan the packet headers for the DSCP values and identify the type of traffic such as data traffic, video traffic, voice traffic. The 6-bit DSCP values classify traffic into 64 classes. However, every value is not considered. The most commonly used are [**53**, **54**]:

(1) Default Forwarding: This is the default per-hop-behavior (PHB). The recommended DSCP value is 0.

(2) Expedited Forwarding: The traffic marked with this value has a low delay, packet loss, and jitter. Voice, video and all the real-time services are marked with this DSCP value. The recommended DSCP value is 46. This traffic is often given strict-priority queuing.

(3) Assured Forwarding: Traffic marked with AF PHB is provided with an assurance of delivery unless congestion occurs. During congestion, the excess traffic beyond the allowed rate is dropped.

7.3.4.2. Metering

Implementing the DSCP values and marking the traffic in SDN is accomplished using metering. OpenFlow 1.3 and later versions support metering [**20**]. Metering is performed

by using a meter table. The meter table is built with multiple meter entries. Each flow can be attached to a meter entry, providing the flexibility to control and impose different operations on the flow. A set of criteria defines every flow. The flows attached to a meter passes through the meter band, where the statistics of the flow is measured such as the rate of flow, number of packets, and packet sizes. Utilizing this information, and with the ability to control the flows, meters can be used to implement QoS.

A traffic monitor application is implemented in the SDNC which receives the traffic statistics from the meter table periodically. The closed-loop feedback control modules implemented in the SDNC use this information in decision making which is described in subsequent sections.

7.3.5. Client

We use the VideoLAN client (VLC) media player [**34**] to play the streaming video and audio content. Multiple clients installed with VLC media players were used to playing the same content from the streaming server.

7.3.6. Server

A standalone media server is used to host all the media content and to stream to all the devices over the network. This server is equipped with an Intel Xenon processor (4 cores) and a 32 GB RAM. We consider three types of video qualities a) 2K Video (High quality), b) 720p (High Definition), and 480p (Low quality).

7.3.7. Mininet

Mininet [**18**] is an open-source network emulator to build a virtual SDN network. It provides a simple and inexpensive way to design and implement a network including many virtual hosts, switches, and controller. The hosts created run standard Linux network software. The switches support OpenFlow for custom routing. This tool is used to simulate the SDN network and the topology as shown in Figure 7.3.

7.3.8. Generation of Traffic

We use VLC media player to broadcast a stream. We select the Real Time Streaming Protocol (RTSP) as the streaming method. We consider three types of video qualities a) 2K (High quality) b) 720p (high definition) and c) 480p (standard definition) and a high-resolution audio (320 Kbps). The chosen video was streamed across the network shown in Fig 5.1. We use the tcpdump packet analyzer to sniff the packets at the server and client's interface, which acts as a sensor. The traffic monitor module collects the settings of the configuration parameters of switches and other network devices present in the data plane in the SDNC. These settings impact the output of the network QoS metrics.

To emulate a real-world attack, we use a DDoS tool, hping3 [**15**]. This tool is capable of generating a large number of UDP and TCP packets. The severity of the attack is varied by varying the packet size. Figure 7.4 shows the attack traffic used in the experiments. This attack traffic can be modeled as a disturbance input into the plant.

7.4. Methodology

7.4.1. Closed Loop Feedback Control

The goal is to enforce the service level agreements and maintain real-time network services. Service level agreement is a commitment promised by the internet service providers to the clients. Service level agreements include many service level objectives (SLO) such as providing a response time of fewer than 2 seconds, network availability, and network latency of less than 30 ms. Another important objective is that the internet service provider cannot afford to drop the 911 emergency calls despite traffic congestion.

In order to reduce unnecessary costs, the service level objectives must be met with using the available resources. To achieve these objectives, I propose two closed-loop feedback control mechanisms as shown in Figure 7.5 and Figure 7.6. These mechanisms guarantee a graceful degradation of real-time services, i.e., to maintain limited functionality of network services even when the network is disrupted by an attack or large amounts of traffic. During an attack, the network services are impacted, and the Quality of Service of the legitimate

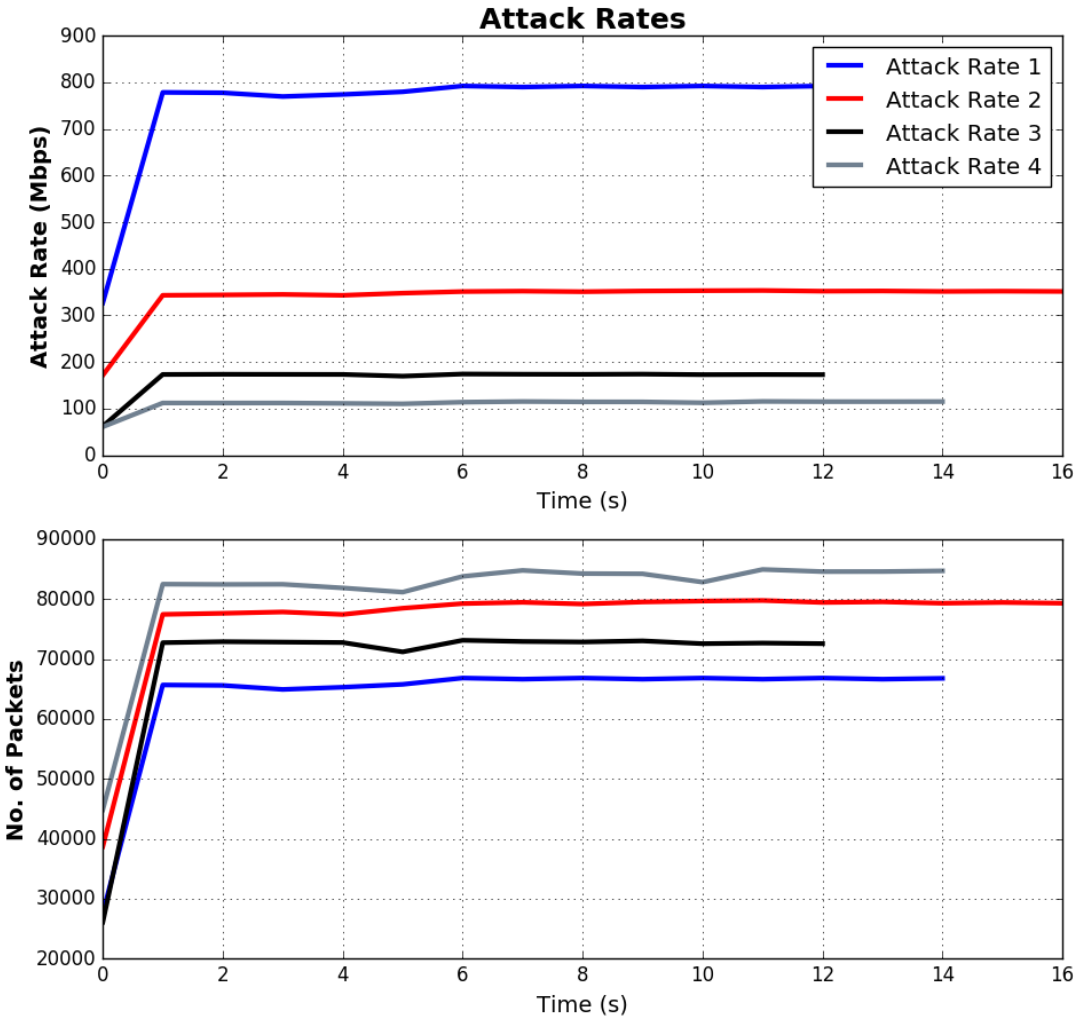

FIGURE 7.4. Attack rates used to emulate the real world DDoS attack (a) rate of packets or bitrate in Mbps, and (b) No. of packets. The attack rates are varied by varying the packet size.

traffic drops drastically. The purpose of the controllers is to limit the impact of an attack, such that the QoS metrics of the network is consistent with those specified in the service level agreements. The service level objectives are provided as a reference input, which is the desired value of the network's output.

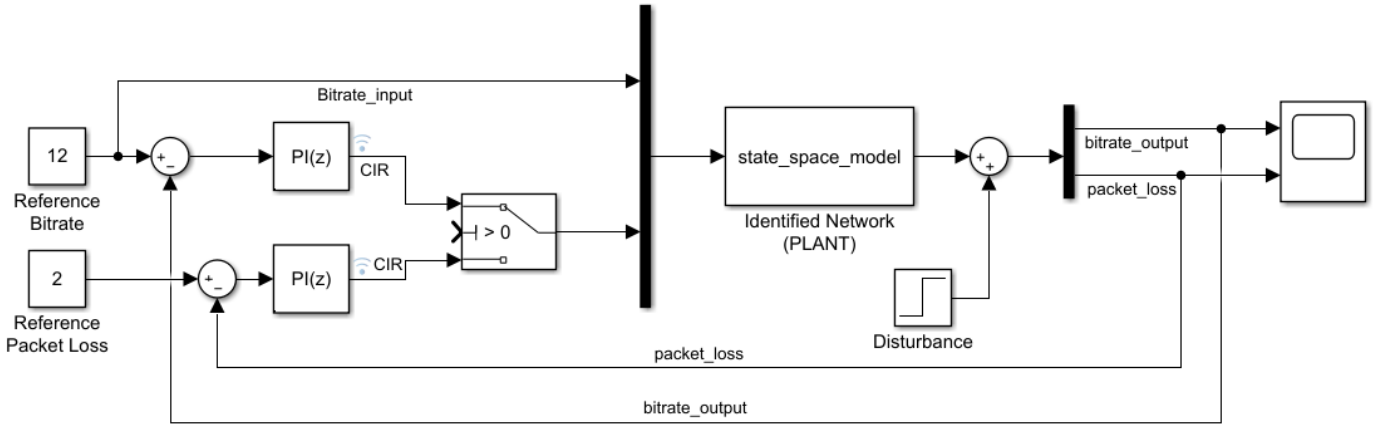

FIGURE 7.5. Closed-loop feedback controller architecture with multi-loop PI controller. The plant is the entire network comprising of all the devices including firewall, routers, clients, and servers. The output from the plant, i.e., the QoS metrics such as bitrate and packet loss are fed back into the multi-loop PI controller. The PI controller designed regulates the process, by adjusting the control input (CIR configuration setting). The switch selects the minimum CIR value to ensure a faster steady-state condition of the network. Bitrate and packet loss are provided as a reference input into the controller. The disturbance represents the DDoS attack which disrupts the QoS.

7.4.2. Committed Information Rate (CIR)

The Committed information rate (CIR) defines the number of tokens removed at each interval. The CIR parameter policies or drops the excess traffic that does not comply with the policy, i.e., if the traffic flow reaches the configured CIR rate, the excess traffic is dropped. The CIR is considered as a control input which regulates the flow of traffic. However, two most important questions arise, i.e.,

(1) How can we dynamically control the traffic rate
(2) What is the appropriate CIR setting to enforce service level objectives.

The answers to these questions lie in designing a suitable controller that offers to

reduce the impact of the attack.

7.4.3. Control Approach 1

The closed-loop feedback architecture designed is shown in Figure 7.5. In this architecture, the multi-loop PI controller is implemented to resolve service disruptions. The controller detects anomalies in the network and regulates the CIR configuration setting to maintain the network in the desired state. From several experiments, it can be inferred that the configuration setting of the CIR affects the bitrate and the packet loss. Hence, each output (bitrate and packet loss) from the system is fed back to an individual PI controller which determine an appropriate CIR value. The minimum value of the controller's output is downloaded into the congested switch to stabilize the network instantly.

The multi-loop PI controller as depicted in 7.5 is a combination of two single-input-single-output (SISO) model. One SISO model quantifies the relation between the bitrate and CIR configuration setting, and the other model captures the relation between the packet loss and CIR. The main objective of any feedback control mechanism is to dynamically control the system or the process to achieve the desired state defined by the reference input. In a PI feedback control mechanism, the measured system output is continuously fed back. The controller receives the control error and determines the appropriate setting of the control input based on the proportional (P) and integral (I) terms. The control error is the difference between the measured output and the reference input.

The general equation of a PI controller is given below:

$$u(t) = K_p e(t) + K_i \int_0^{t'} e(t')dt' \tag{5}$$

The proportional (K_p) variable adjusts the system output proportionally to the error signal by controlling the proportional gain of the controller. Isolated use of a proportional controller might help in reducing the error, but the final output might result in oscillations such as an on-off signal. These oscillations can be reduced by the integral variable (K_i).

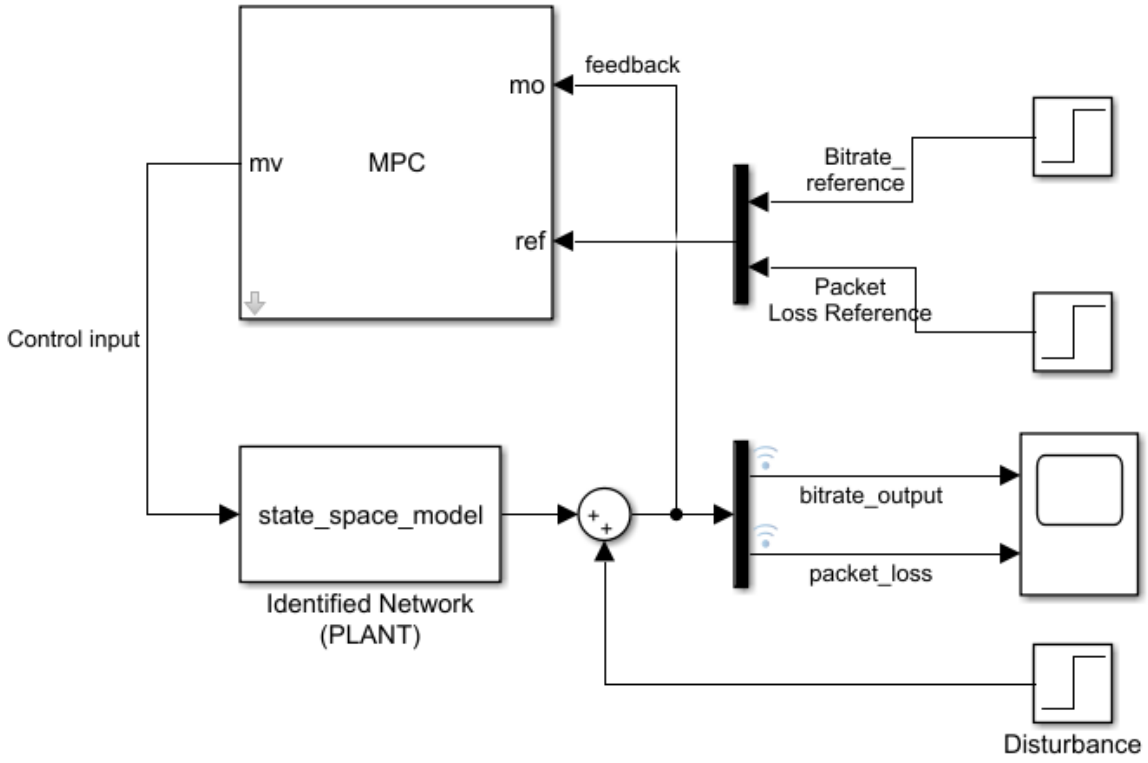

FIGURE 7.6. The closed-loop feedback controller architecture with MPC controller. The plant is the entire network comprising of all the devices including firewall, routers, clients, and servers. The output of the plant which is the QoS metrics of the real-time services is fed to the model through mo. mv is the manipulated variable or the control input which is predicted by the controller. Bitrate and packet loss are provided as a reference input.

7.4.4. Control Approach 2

Figure 7.6 shows the model predictive controller which constitutes the closed-loop feedback architecture implemented in this approach. A model predictive controller [**48**] is an excellent feedback control algorithm that uses a model to predict the future outputs of a process. The controller achieves this by using an explicit dynamic model of the system response to the *manipulated variables* (MVs) or the control input. The controller uses

this model to control the plant. MPC controller has been applied to many process control industries including chemical plants, oil refineries, and power plants. MPC has been proven to be a better option to PI controller for complex systems [**48**]. Unlike a PI controller, an MPC controller provides the flexibility of controlling multiple system configurations which impact the system outputs.

The MPC controller uses the model of the plant to predict the future plant output behavior. The *prediction horizon* determines the number of future control intervals the controller must predict. The controller uses an optimizer to ensure the predicted output follows the desired reference input. It minimizes the error of the predicted output and the reference input by solving the online optimization problem at every control interval. The solution regulates the control input or the manipulated variables to be used in the plant until the next interval.

Figure 2.4 shows the anatomy of the closed loop feedback architecture using the MPC controller. The MPC controller is integrated into the SDN controller. It utilizes the traffic statistics from the traffic monitor module of the SDNC. The traffic monitor feeds the network characteristics to the MPC controller. The controller detects the abnormalities of the network service and adjusts the control input to guarantee the service quality. This control input is downloaded into the congested switch in the data plane. The control input is the CIR configuration setting which drops the excess traffic when the traffic reaches the configured rate. The congested switch is identified through link utilization by the SDN controller.

The state-space model of the plant contained inside the MPC controller is a second order discrete-time state space model shown below.

$$x(t+Ts) = Ax(t) + Bu(t) + Ke(t)$$
$$y(t) = Cx(t) + Du(t) + e(t) \quad (6)$$

Where u is the input, x is the state, y is the output and e is the error. A, B, C, D, and K are matrix coefficients and must have these characteristics:

A must be an n-by-n matrix, where n is the number of states.

B must be an n-by-m matrix, where m is the number of inputs.

C must be an r-by-n matrix, where r is the number of outputs.

D must be an r-by-m matrix.

I use the the system identification toolbox of matlab [**28**] to construct the state-space model. The following are the parameters estimated using the toolbox:

$$A = \begin{bmatrix} 0.7359 & -0.3647 & -0.0143 & -0.0604 \\ -0.7647 & -0.3437 & -0.3779 & -0.3745 \\ 1.1346 & 1.0939 & 0.5318 & -0.8582 \\ -0.0674 & 0.1281 & 0.4638 & 0.4950 \end{bmatrix}$$

$$B = \begin{bmatrix} 0.0019 & 0.0329 \\ 0.0098 & 0.1202 \\ 0.0010 & -0.0554 \\ 0.0003 & 0.0314 \end{bmatrix}$$

$$C = \begin{bmatrix} 125.5362 & -33.3935 & 9.3692 & 0.4002 \\ -12.3948 & 2.9304 & -0.1076 & -0.5324 \end{bmatrix}$$

$$D = \begin{bmatrix} 3.1330 & 9.2601 \\ -0.2920 & 0.0879 \end{bmatrix}$$

$$K = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

7.4.4.1. Tuning

Tuning an MPC controller is to minimize the dynamic control objective or the cost function. The dynamic control objective is a mathematical optimization problem which selects the "best available" values from the available alternatives. Solving the optimization problem involves minimizing or maximizing the cost function to find a best possible solution for the controller. The objective function implemented in the MPC controller is L_1-norm objective. It is the absolute value of the difference between the current measured control value and the desired value [**70**]. The function is shown below:

$$\min_{x,y,u} \Phi = w_{hi}^T e_{hi} + w_{lo}^T e_{lo} + y^T c_y + u^T c_u + \Delta u^T c_{\Delta u}$$

$$\text{s.t.} \quad 0 = f\left(\frac{dx}{dt}, x, y, p, d, u\right)$$

$$0 = g(x, y, p, d, u)$$

$$0 \leq h(x, y, p, d, u)$$

$$\tau_c \frac{dy_{t,hi}}{dt} + y_{t,hi} = sp_{hi}$$

$$\tau_c \frac{dy_{t,lo}}{dt} + y_{t,lo} = sp_{lo}$$

$$e_{hi} \geq y - y_{t,hi}$$

$$e_{lo} \geq y_{t,lo} - y$$

Φ	objective function
y	model values $(y_0, \ldots, y_n)^T$
$y_t, y_{t,hi}, y_{t,lo}$	desired trajectory target or dead-band
w_{hi}, w_{lo}	penalty outside trajectory dead-band
$c_y, c_u, c_{\Delta u}$	cost of y, u and Δu, respectively
u, x, p, d	inputs (u), states (x), parameters (p), and disturbances (d)
f, g, h	equation residuals (f), output function (g), and inequality constraints (h)
τ_c	time constant of desired controlled variable response
e_{lo}, e_{hi}	slack variable below or above the trajectory dead-band
sp, sp_{lo}, sp_{hi}	target, lower, and upper bounds to final setpoint dead-band

7.5. Results and Analysis

I implement the SDN network as shown in Figure 7.3 to study the performance of both the feedback architectures. The switches in the data plane are connected in a mesh topology. These switches are connected to the controller, that has the ability to control the

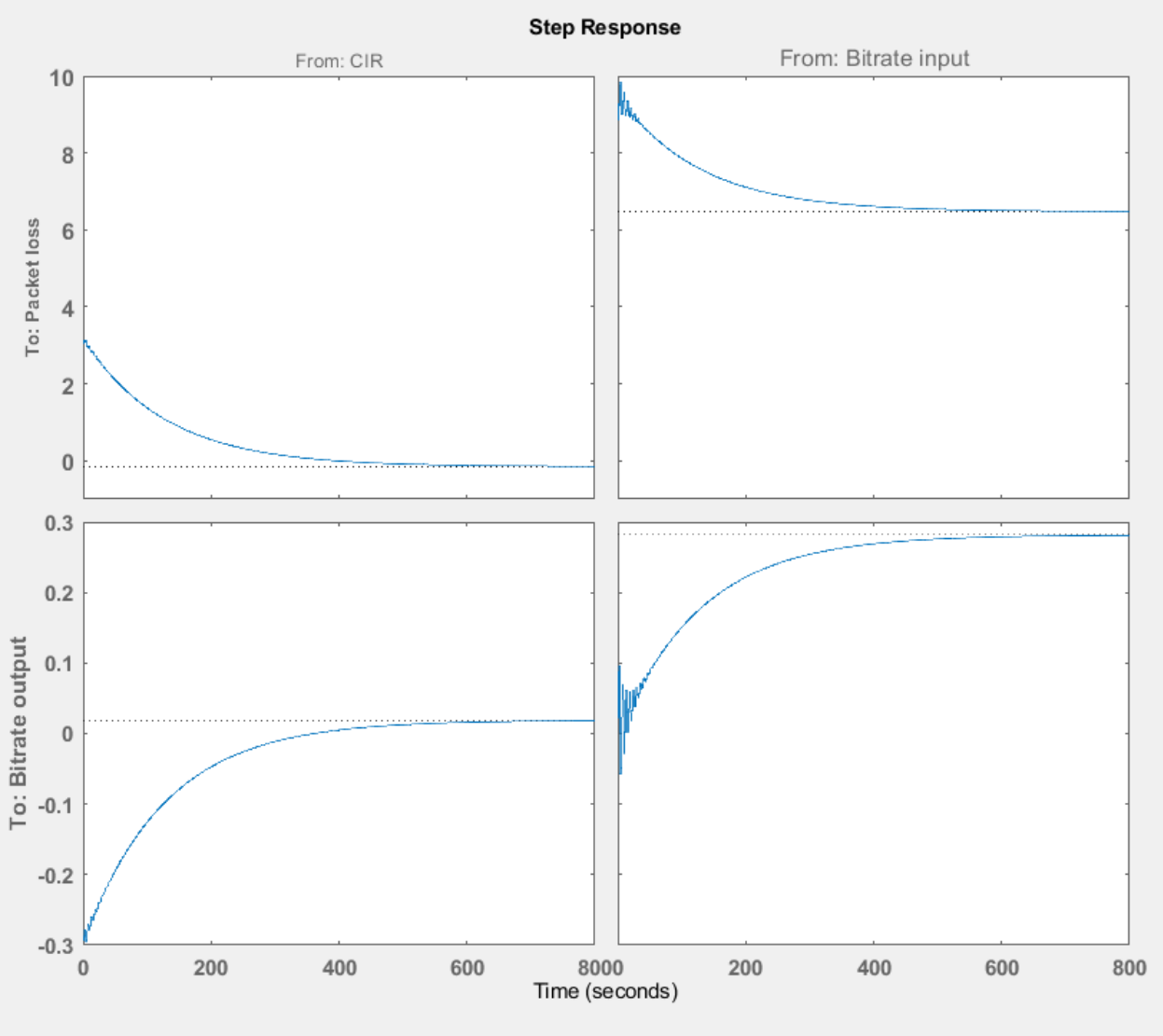

FIGURE 7.7. The unit step response of both the outputs w.r.t the control inputs of the plant model contained in the MPC controller. The plant is a second order state-space system. The oscillations in the response reveal the under-damped dynamics of the plant.

flow, configure the switch and monitor the traffic, using OpenFlow communication protocol. The data plane is set up using mininet, and the Ryu controller [**23**] is used as an SDN controller (SDNC). The switches update their flow statistics to the controller periodically. The PI controller and the MPC controller are implemented as a module in the SDNC. Traffic statistics such as a number of flows, packets per second, a rate of packets in bytes per second,

are fed back to the controllers.

The audio and video traffic is streamed across the network using the VLC streaming server. The clients play the video using VLC client application. Both the audio and video traffic are marked as high priority, i.e., DSCP=46. Three different types of video streaming resolutions are considered i) 2k Video ii) 720p Video (high definition) and a iii) 480p Video (standard definition). The video played contains many fast moving objects, explaining the reason for a varying bitrate, while, the bitrate of the audio stream is constant.

In order to emulate a real-world DDoS attack, the network is flooded with a large number of packets as shown in Figure 7.4. The attack resembles a step input which is modeled as disturbance into the plant. When the network is under attack, the links get congested resulting in dropping of packets, increasing the delay and jitter of the services 7.8. The QoS of the network service drops drastically. Despite prioritizing both video and audio traffic, the attacks impact the high priority traffic as shown in Figure 7.8 and 7.9.

7.5.1. PI Controller

As mentioned earlier in Section 7.4.3, the PI controller is designed as a multi-loop SISO controller as shown in Figure 7.5. The controllers are connected to a switch which selects the minimum CIR value (control input), stabilizing the network. The output of the PI controller implemented is shown in Figure 7.8 and 7.9. The QoS of the network services, which is the output of the system, is fed continuously into the controller. When the network is under attack, the QoS of the real-time traffic drops drastically. The PI controller detects the abnormalities in the network characteristics from the feedback signal. The controller adjusts the CIR input to the system calculated from the error signal making the network resilient towards the attack. The CIR input provided by the controller controls the data/attack traffic.

Figure 7.8 presents the output from the network shown in Figure 7.3, collected from the PI controller scenario. A 720p resolution video is streamed across the SDN network. The variation in the bitrate is due to the content of the video containing many fast moving objects. The QoS metrics of the video, collected from the traffic monitor module present in

the SDNC 7.3 is fed to the controller. After nearly 40 seconds, the network is flooded with a significant amount of UDP packets, congesting the links. Due to the congested links, a large number of packets are dropped. The packet loss of the video traffic sharply increased to nearly 50% and the bitrate of the video drops to nearly 50 kbps, degrading the quality of the video. The PI controller detects this drop in the QoS of the video traffic and adjusts the control input, i.e., the CIR value of the congested node to bring back the network to the desired state.

The setpoint values or the reference values define the desired state. The controller tries to match the measured system output to the provided reference values by adjusting the CIR input. Since PI controller is capable of controlling single input and single output system, each PI controller try to reduce the CIR value based on the corresponding reference input. The reference values of bitrate and packet loss are set to 7 Mbps and 3% respectively, i.e., the bitrate of the video should be maintained at 7 Mbps, and the network can afford up to 3% packet loss. These values are based on the SLA agreements of the network.

After nearly 60 seconds, the controller achieves its goal of stabilizing the traffic by minimizing the control error which is the difference between the current measured output value and the reference input. The currently measured output of the system is fed periodically to the controller. The SDNC provides the flexibility to the PI controller to dynamically set the CIR value. The PI controller gradually decreases the CIR value until the measured system output converges to the reference value. The video is restored back at t=98s when the control objectives are met. The packet loss drops down to less than 5%, and the bitrate of the video matches the reference value. The settling time which is measured as the period from the detection of the disruption in the video traffic to bringing the system to the desired state is nearly 60 seconds.

Figure 7.9 shows the QoS metrics of the network traffic collected while streaming high definition audio. From the figure, it is evident that the bitrate of the audio traffic is very smooth, unlike the video traffic. The bitrate of the audio traffic is much lower than the video traffic. During an attack, the packet loss increases to nearly 70%. The controller performs

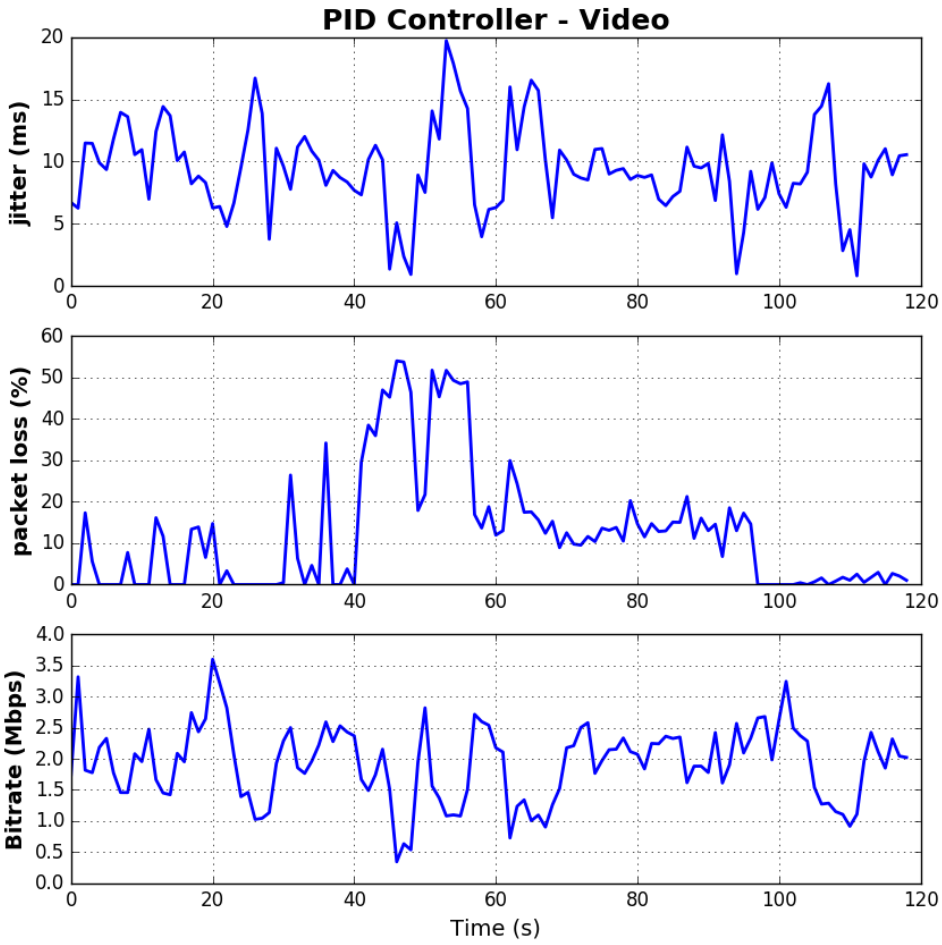

FIGURE 7.8. (a) Jitter, (b) packet loss and (c) bitrate measured from the SDN network under attack. At time T=40 seconds, an attack causes a drop in QoS by nearly 50% causing severe deterioration of the video. The percentage of packet loss increased to 50%, and the bitrate of the video dropped to 50 KBPS from 250 KBPS, which is unacceptable. The controller continuously receives the feedback and manipulates the CIR configuration setting. The impact of the attack is reduced after nearly 60 seconds. This delay is due to the functionality of the PI controller, tuned as a SISO operation.

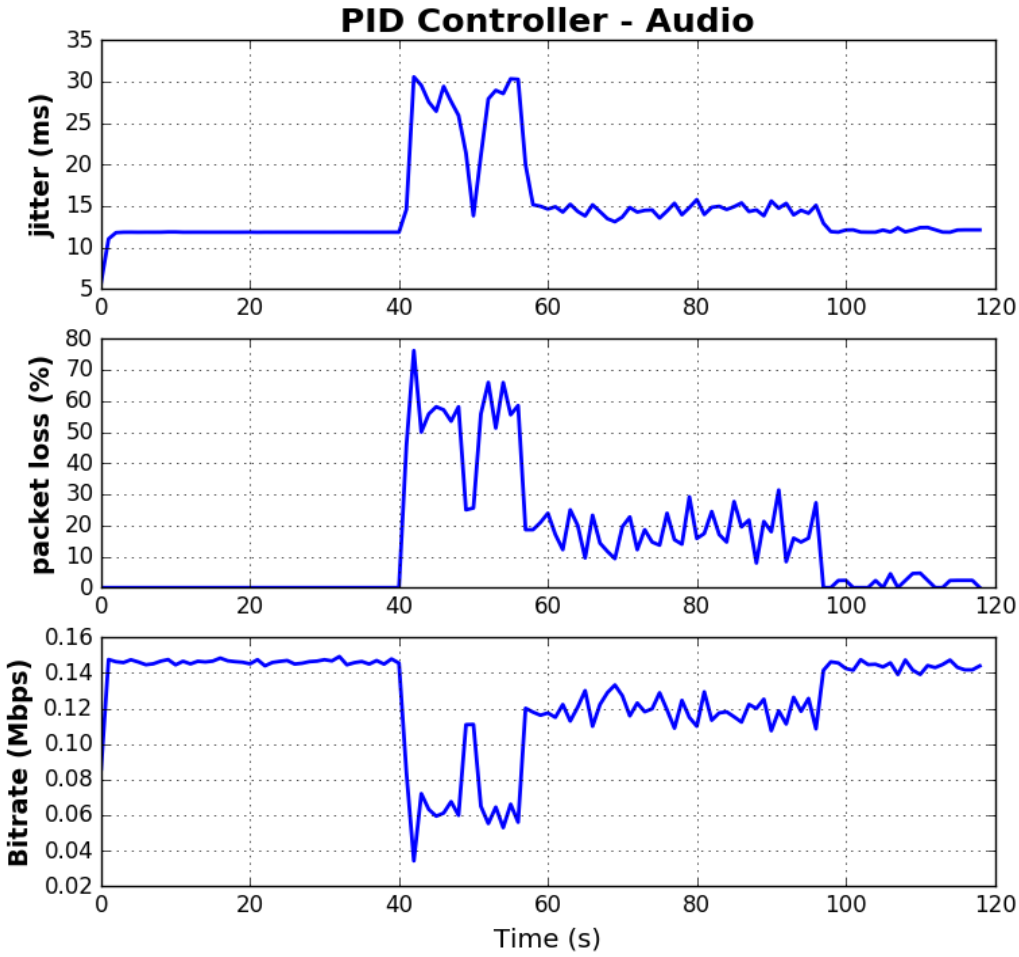

FIGURE 7.9. (a) Jitter, (b) packet loss and (c) bitrate measured from the SDN network under attack. At time T=40 seconds, an attack causes a drop in QoS by nearly 70% causing severe deterioration of the audio. The controller continuously receives the feedback and manipulates the CIR configuration setting. The impact of the attack is reduced after nearly 60 seconds. This delay is due to the functionality of the PI controller, tuned as a SISO operation.

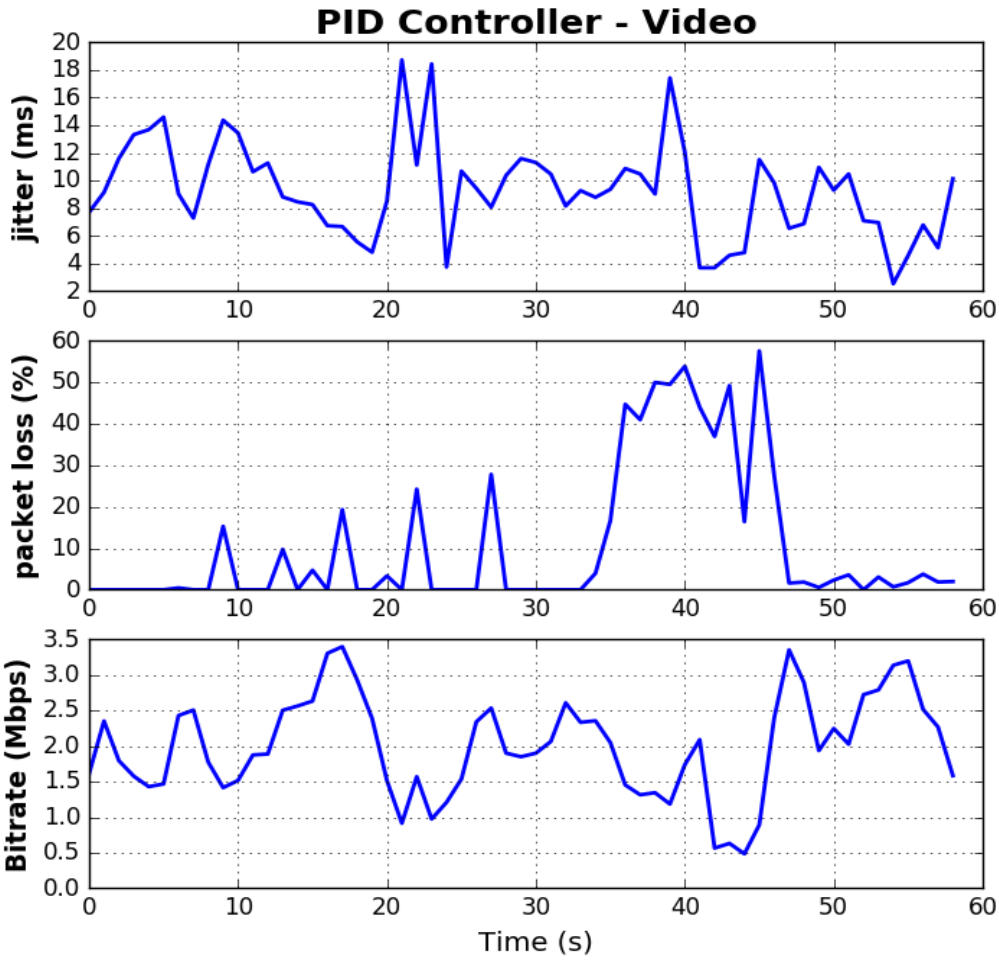

FIGURE 7.10. (a) Jitter, (b) packet loss and (c) bitrate measured from the SDN network under attack. An attack is simulated at time t=30 seconds. When the network is under attack, the percentage of packet loss increased to 50%, and the bitrate of the video dropped to 500 KBPS from 3 Mbps. As a result, the QoS of the real-time traffic dropped by nearly 50% causing severe deterioration of the video. The MPC controller detects the drop in the QoS and adjusts the Committed information rate (CIR) to reduce the impact of an attack and restore the services. The controller regulates the process by adjusting the controlled input (CIR) to achieve the desired QoS metrics (bitrate=250KBPS and packet loss less than 5%). The video stream is restored after nearly 20 seconds.

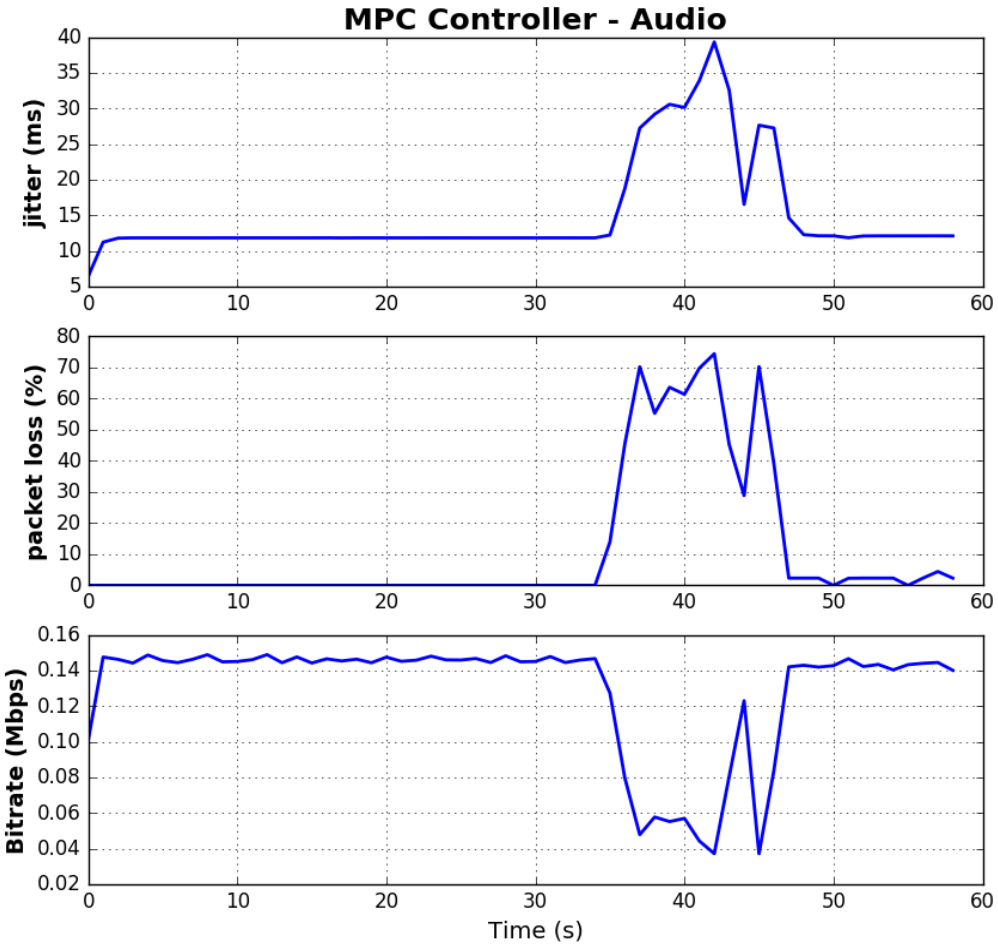

FIGURE 7.11. (a) Jitter, (b) packet loss and (c) bitrate measured from the SDN network under attack. An attack is simulated at time t=35 seconds. When the network is under attack, the percentage of packet loss peaked at 70%, and the bitrate of the audio dropped to 20 KBPS from 150 KBPS, causing severe deterioration of the audio. The MPC controller detects the drop in the QoS and adjusts the Committed information rate (CIR) to reduce the impact of an attack and restore the services. The controller regulates the process by adjusting the controlled input (CIR) to achieve the desired QoS metrics (bitrate=250KBPS and packet loss less than 5%). The audio stream is restored after nearly 20 seconds.

a similar functionality in controlling the degradation of audio.

7.5.2. MPC Controller

The MPC controller is an advanced multivariable controller which provides the flexibility of controlling a MIMO system. The controller implemented in this scenario controls the network under attack by adjusting the CIR value, inspecting both the measured system outputs simultaneously. From the experiments conducted it is observed that the both the CIR value setting and the input bit rate of the video affect the behavior of the network. With MPC's inherent property of being able to control a MIMO system provides an accurate prediction of the CIR configuration parameter.

Figure 7.10 shows the output of the MPC controller architecture implemented while streaming a high definition video stream (720p). This data represents the QoS metrics of the video traffic collected every second from the client. The reference values of the bitrate and the packet loss are set to 7 Mbps and 3% respectively. After 40 seconds, the network is flooded with a large number of packets, congesting the network. Consequently, the QoS of the video traffic drops to nearly 50%, increasing the control error. This control error is used to solve the online optimization problem by the MPC controller, and the solution determines the setting of the control input (CIR value). As depicted in the figure, the impact of the attack on the video traffic is limited after nearly 20 seconds. Unlike the PI controller, the MPC controller quickly restores the quality of the video.

A similar approach was applied to the network while streaming high definition audio. The output of the QoS metrics of the audio stream is shown in 7.11. The quality of the audio is restored in nearly 15 seconds.

Figure 7.12 shows the predicted control input (CIR values) of the designed MPC controller for various resolutions of video. The congested switches in the data plane are configured with the predicted CIR values periodically. We can observe that the predicted value is inversely proportional to the bitrate of the video. The higher the bitrate of the video, lower is the predicted CIR value. Initially, CIR is set to 100. When the network is disrupted by an attack, the QoS metrics drop thereby increasing the control error. This

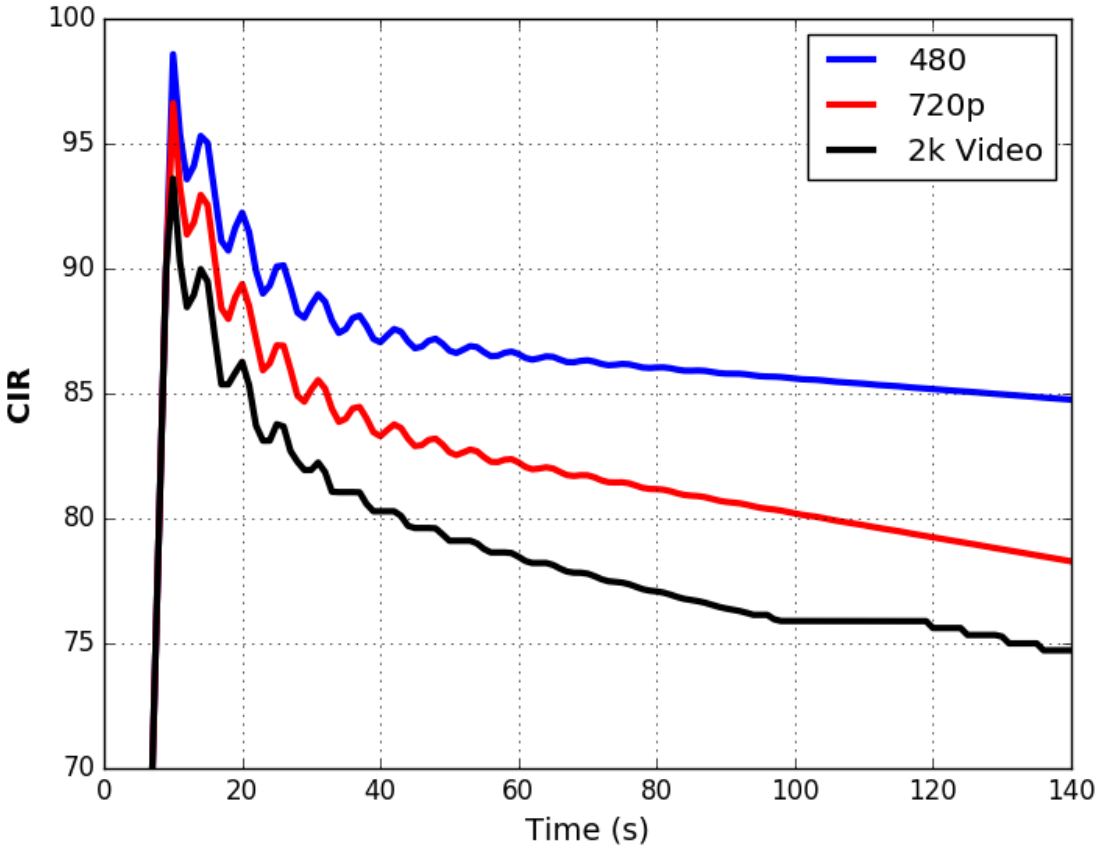

FIGURE 7.12. The predicted control input (CIR values) of the designed MPC controller for various resolutions of video. The blue, red and black lines are the predicted CIR configuration settings for 2k (10 Mbps), 720p (7 Mbps) and 480p(5 Mbps) video resolutions respectively. The predicted CIR values are inversely proportional to the bitrate of the video. The predicted CIR values reveal the under-damped dynamics of the controller. After nearly 40 seconds, the oscillating CIR values begin to converge.

control error is used by the plant model contained in the MPC architecture to predict the CIR values. Furthermore, it can be inferred from the figure that the predicted CIR values reveal the under-damped dynamics of the controller. After nearly 40 seconds, the oscillating CIR values begin to converge to a steady state.

Since attack traffic is not detected, the dropped traffic might include insignificant data traffic. This methodology holds good for graceful degradation of the real-time services and the network. This approach can be applied to mission-critical real-time applications such as surveillance traffic, 911 emergency calls and air traffic control, where packet loss cannot be tolerated. Most streaming video applications use TCP as the underlying transport protocol, which intuitively handles network congestion.

The proposed approached steers the network towards desired performance as prescribed by the SLAs. It is important to note that the controllers designed are modeled for disturbance rejection. The limitation of a PI controller is that the controller has a longer settling time to perform the control actions, as evidenced by the results. Whereas, using the MPC controller the QoS of real-time traffic gracefully degrades the network within a shorter settling time. MPC controller has the flexibility of handling Multi-input-multi-output systems where multi-variable inputs control the outputs. On the other hand, PI controllers handle only SISO systems. However, PI controllers can process MIMO controller by using multiloop controllers as previously described in Section 7.4.3 but this approach is not scalable.

This methodology scales well in response to any magnitude of the DDoS attack. UDP flooding was considered as a significant source for the DDoS attack. However, this principle also holds good for TCP based DDoS attacks such as SYN flooding, where the controller would tune the burst size parameter to minimize the number of TCP connections.

CHAPTER 8

CONCLUSION

Network services are still vulnerable to DDoS attacks. The recent massive attacks have proved the fact that attackers are finding new and innovative ways to attack and destroy a network or a target server. With the rise of artificial intelligence, attackers could use new ideas and novel methods to generate attacks which could be more powerful and destructive. The security tools and the devices should be implemented with a foresight of the damage caused by these attacks.

This dissertation focuses on employing control theoretic approaches to reduce the impact of any volumetric attack. I designed online system identification models and implemented dynamic and predictive feedback control mechanism to analyze the stability of the network and to ensure graceful degradation of network services in the face of an attack. The mechanism understands the network dynamics in any conditions and provides system resilience in the face of an attack. The feedback mechanisms employed to stabilize the unstable network in real-time.

I would like to emphasize the fact that the proposed methodology does not detect the attack. The impact of an attack is detected by the measuring the rate of degradation of the network services. The proposed feedback mechanisms were more effective in SDN network than a conventional network. The centralized nature of SDN enabled increased network visibility and controllability, which is desirable for control systems. Whereas, in conventional networks, a network-aware centralized controller is required. These methodologies scale well in response to any magnitude of the DDoS attack. Volumetric attacks were considered as a significant source for the DDoS attack throughout the dissertation.

I implemented three types of control algorithms. However, the closed-loop feedback controller can also be implemented using other efficient controllers such as a Fuzzy logic controller which uses heuristic rules to determine the action of the controller. Advanced methods such as neural networks [**41**, **68**, **69**, **83**] can also be employed to perform accurate

system identification and efficient control.

8.1. Applicability of Presented Techniques

The solutions discovered can be applied to real-time services, primarily to mission critical applications where the threshold for packet loss is very low.

The PI and the MPC controllers can be deployed in any network device of a network such as a firewall or a router. It can also be included as a module inside an SDN controller as mentioned in chapter 6.

8.2. Limitations

- The feedback control approach implemented in Chapter 5 and 6 holds good only for real-time services. This approach is not suitable for an attack on data traffic, as the mechanism is only capable of identifying prioritized traffic and prioritizing non-real-time traffic is not an acceptable method. However, the methodology discussed in Chapter 3 and 4, i.e., implementing parallel links can be applied to all types of traffic.
- If the details of the configuration settings of the trusted node are revealed to the attacker, this methodology might not function as expected.

REFERENCES

[1] *Apache jmeter - apache jmeter^{TM}*, http://jmeter.apache.org/index.html, (Accessed on 07/12/2018).

[2] *Black-box modeling - matlab & simulink*, https://www.mathworks.com/help/ident/ug/black-box-modeling.html, (Accessed on 06/21/2018).

[3] *Bots, botnets and zombies — nacha*, https://www.nacha.org/news/bots-botnets-and-zombies, (Accessed on 07/13/2018).

[4] *A brief history of ddos attacks. — nota bene: Eugene kaspersky's official blog*, https://eugene.kaspersky.com/2016/12/06/a-brief-history-of-ddos-attacks/, (Accessed on 07/12/2018).

[5] *Chaos engineering: Why the world needs more resilient systems*, https://www.infoq.com/presentations/chaos-engineering-resilient-systems, (Accessed on 07/12/2018).

[6] *Cyberwar - a threat to business*, http://www.gideonrasmussen.com/article-14.html, (Accessed on 07/13/2018).

[7] *Ddos attack hits dyn dns servers, knocks twitter, spotify, reddit offline*, https://www.dailydot.com/layer8/dyn-ddos-twitter-reddit-east-coast/, (Accessed on 07/12/2018).

[8] *Ddos attack trends*, https://www.verisign.com/assets/infographic-ddos-trends-Q42017.pdf, (Accessed on 06/21/2018).

[9] *Ddos attacks increased 91% in 2017 thanks to iot - techrepublic*, https://www.techrepublic.com/article/ddos-attacks-increased-91-in-2017-thanks-to-iot/, (Accessed on 06/21/2018).

[10] *Ddos breach costs rise to over $2m for enterprises finds kaspersky lab report — kaspersky lab us*, https://usa.kaspersky.com/about/press-releases/2018_

ddos-breach-costs-rise-to-over-2m-for-enterprises-finds-kaspersky-lab-report, (Accessed on 07/13/2018).

[11] *Dyn analysis summary of friday october 21 attack — dyn blog*, https://dyn.com/blog/dyn-analysis-summary-of-friday-october-21-attack/, (Accessed on 07/12/2018).

[12] *Dynamic control tuning*, http://apmonitor.com/do/index.php/Main/ControllerTuning, (Accessed on 07/11/2018).

[13] *Electricity grid in u.s. penetrated by spies - wsj*, https://www.wsj.com/articles/SB123914805204099085, (Accessed on 07/13/2018).

[14] *Github hit with the largest ddos attack ever seen — zdnet*, https://www.zdnet.com/article/github-was-hit-with-the-largest-ddos-attack-ever-seen/, (Accessed on 06/21/2018).

[15] *hping security tool - hping3 information*, http://www.hping.org/hping3.html, (Accessed on 07/12/2018).

[16] *Linux containers*, https://linuxcontainers.org/, (Accessed on 07/12/2018).

[17] *Major ddos attack on dyn dns knocks spotify, twitter, github, paypal, and more offline — pcworld*, https://www.pcworld.com/article/3133847/internet/ddos-attack-on-dyn-knocks-spotify-twitter-github-etsy-and-more-offline.html, (Accessed on 07/12/2018).

[18] *Mininet: An instant virtual network on your laptop (or other pc) - mininet*, http://mininet.org/, (Accessed on 07/13/2018).

[19] *Modern updates in pid control tuning — control engineering*, https://www.controleng.com/single-article/modern-updates-in-pid-control-tuning.html, (Accessed on 07/13/2018).

[20] *Open networking foundation is an operator led consortium leveraging sdn, nfv and cloud technologies to transform operator networks and business models*, https://www.opennetworking.org/, (Accessed on 07/13/2018).

[21] *Openflow ¡ pertkb ¡ twiki*, https://kb.pert.geant.net/PERTKB/OpenFlow, (Accessed on 07/13/2018).

[22] *The real impact of the ddos against dyn - turbobytes*, https://www.turbobytes.com/blog/real-impact-of-ddos-against-dyn/, (Accessed on 07/13/2018).

[23] *Ryu sdn framework*, https://osrg.github.io/ryu/, (Accessed on 07/13/2018).

[24] *Selecting a model structure in the system identification process - national instruments*, http://www.ni.com/white-paper/4028/en/, (Accessed on 05/20/2018).

[25] *Skype suffering connectivity issues, allegedly due to ddos attack*, https://www.scmagazineuk.com/skype-suffering-connectivity-issues-allegedly-due-to-ddos-attack/article/670071/, (Accessed on 06/21/2018).

[26] *Software-defined networking (sdn) definition - open networking foundation*, https://www.opennetworking.org/sdn-definition/, (Accessed on 07/13/2018).

[27] *The stride threat model — microsoft docs*, https://docs.microsoft.com/en-us/previous-versions/commerce-server/ee823878(v=cs.20), (Accessed on 07/11/2018).

[28] *System identification toolbox - matlab*, https://www.mathworks.com/products/sysid.html, (Accessed on 07/09/2018).

[29] *Tag: slow http attack — qualys blog*, https://blog.qualys.com/tag/slow-http-attack, (Accessed on 07/13/2018).

[30] *Three ways ddos attacks are evolving in 2017 — liquid web*, https://www.liquidweb.com/blog/three-ways-ddos-attacks-evolving-2017/, (Accessed on 05/20/2018).

[31] *Tr10: Software-defined networking - mit technology review*, http://www2.technologyreview.com/news/412194/tr10-software-defined-networking/, (Accessed on 07/12/2018).

[32] *Understanding qos policing and marking on the catalyst 3550 - cisco*, https://www.cisco.com/c/en/us/support/docs/switches/catalyst-3550-series-switches/24800-153.html, (Accessed on 05/20/2018).

[33] *Use perfmon to diagnose common server performance problems*, https://technet.microsoft.com/en-us/library/2008.08.pulse.aspx, (Accessed on 07/13/2018).

[34] *Vlc: Official site - free multimedia solutions for all os! - videolan*, https://www.videolan.org/index.html, (Accessed on 05/20/2018).

[35] *Welcome! - the apache http server project*, https://httpd.apache.org/, (Accessed on 07/12/2018).

[36] *What are bots, botnets and zombies? — webroot*, https://www.webroot.com/us/en/resources/tips-articles/what-are-bots-botnets-and-zombies, (Accessed on 07/13/2018).

[37] *What is a ddos attack? see how ddos attacks can impact your business - verisign*, https://www.verisign.com/en_US/security-services/ddos-protection/what-is-a-ddos-attack/index.xhtml, (Accessed on 07/12/2018).

[38] *Super bowl xlix: Tom brady vs. russell wilson — patriots vs. seahawks — nfl full game - youtube*, https://www.youtube.com/watch?v=0RFXLwZV_fA&t=2649s, september 2016, (Accessed on 07/15/2018).

[39] Eitan Altman, Aniruddha Singhal, Corinne Touati, and Jie Li, *Resilience of routing in parallel link networks*, International Conference on Decision and Game Theory for Security, Springer, 2016, pp. 3–17.

[40] Kiam Heong Ang, Gregory Chong, and Yun Li, *Pid control system analysis, design, and technology*, IEEE transactions on control systems technology 13 (2005), no. 4, 559–576.

[41] Panos J Antsaklis et al., *Neural networks for control systems*.

[42] Dharma Aryani, Liuping Wang, and Tharindu Patikirikorala, *Control oriented system identification for performance management in virtualized software system*, IFAC Proceedings Volumes 47 (2014), no. 3, 4122–4127.

[43] Karl Johan Åström, Tore Hägglund, and Karl J Astrom, *Advanced pid control*, (2006).

[44] Aman Bakshi and Yogesh B Dujodwala, *Securing cloud from ddos attacks using in-*

trusion detection system in virtual machine, Communication Software and Networks, 2010. ICCSN'10. Second International Conference on, IEEE, 2010, pp. 260–264.

[45] Christos G Bampis, Zhi Li, and Alan C Bovik, *Continuous prediction of streaming video qoe using dynamic networks*, IEEE Signal Processing Letters 24 (2017), no. 7, 1083–1087.

[46] Eric D Bankaitis, Matthew E Bechard, and Christopher VE Wright, *Feedback control of growth, differentiation, and morphogenesis of pancreatic endocrine progenitors in an epithelial plexus niche*, Genes & development 29 (2015), no. 20, 2203–2216.

[47] Arthur Earl Bryson, *Applied optimal control: optimization, estimation and control*, Routledge, 2018.

[48] Eduardo F Camacho and Carlos Bordons Alba, *Model predictive control*, Springer Science & Business Media, 2013.

[49] Chao Chen, Lark Kwon Choi, Gustavo de Veciana, Constantine Caramanis, Robert W Heath, and Alan C Bovik, *Modeling the time—varying subjective quality of http video streams with rate adaptations*, IEEE Transactions on Image Processing 23 (2014), no. 5, 2206–2221.

[50] Zhijiang Chen, Guobin Xu, Vivek Mahalingam, Linqiang Ge, James Nguyen, Wei Yu, and Chao Lu, *A cloud computing based network monitoring and threat detection system for critical infrastructures*, Big Data Research 3 (2016), 10–23.

[51] Kevin Coleman, *Ddos attacks grow with a new twist! — 2017-07-10 — security magazine*, `https://www.securitymagazine.com/blogs/14-security-blog/post/88141-ddos-attacks-grow-with-a-new-twist`, (Accessed on 05/20/2018).

[52] Ram Dantu, Joao W Cangussu, and Sudeep Patwardhan, *Fast worm containment using feedback control*, IEEE Transactions on Dependable and Secure Computing 4 (2007), no. 2, 119–136.

[53] Bruce Davie, Anna Charny, JCR Bennet, Kent Benson, Jean-Yves Le Boudec, William Courtney, Shahram Davari, Victor Firoiu, and Dimitrios Stiliadis, *An expedited forwarding phb (per-hop behavior)*, Tech. report, 2002.

[54] Elwyn Davies, Mark A Carlson, Walter Weiss, David Black, Steven Blake, and Zheng Wang, *An architecture for differentiated services*, (1998).

[55] Luca De Cicco and Saverio Mascolo, *An experimental investigation of the akamai adaptive video streaming*, Symposium of the Austrian HCI and Usability Engineering Group, Springer, 2010, pp. 447–464.

[56] ———, *A mathematical model of the skype voip congestion control algorithm*, IEEE Transactions on Automatic Control 55 (2010), no. 3, 790–795.

[57] ———, *An adaptive video streaming control system: Modeling, validation, and performance evaluation*, IEEE/ACM Transactions on Networking (TON) 22 (2014), no. 2, 526–539.

[58] Luca De Cicco, Saverio Mascolo, and Vittorio Palmisano, *Feedback control for adaptive live video streaming*, Proceedings of the second annual ACM conference on Multimedia systems, ACM, 2011, pp. 145–156.

[59] Clarence W De Silva, *Intelligent control: fuzzy logic applications*, CRC press, 2018.

[60] John C Doyle, Bruce A Francis, and Allen R Tannenbaum, *Feedback control theory*, Courier Corporation, 2013.

[61] M Efe et al., *Neural network assisted computationally simple pid control of a quad rotor uav*, IEEE Trans. on Industrial Informatics 7 (2011), no. 2, 354–361.

[62] Gerhard P Fettweis, *The tactile internet: Applications and challenges*, IEEE Vehicular Technology Magazine 9 (2014), no. 1, 64–70.

[63] Antonio Filieri, Martina Maggio, Konstantinos Angelopoulos, Nicolás D'Ippolito, Ilias Gerostathopoulos, Andreas Berndt Hempel, Henry Hoffmann, Pooyan Jamshidi, Evangelia Kalyvianaki, Cristian Klein, et al., *Software engineering meets control theory*, Proceedings of the 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, IEEE Press, 2015, pp. 71–82.

[64] Paul L Frankhouser, Maureen L Mulvihill, Roger B Bagwell, Ryan S Clement, Gabriela Hernandez Mesa, Ryan M Sheehan, and Brian M Park, *Medical tool with electromechanical control and feedback*, December 29 2015, US Patent 9,220,483.

[65] Anteneh Girma, Moses Garuba, Jiang Li, and Chunmei Liu, *Analysis of ddos attacks and an introduction of a hybrid statistical model to detect ddos attacks on cloud computing environment*, Information Technology-New Generations (ITNG), 2015 12th International Conference on, IEEE, 2015, pp. 212–217.

[66] Jesus M Gonzalez, Mohd Anwar, and James BD Joshi, *A trust-based approach against ip-spoofing attacks*, Privacy, Security and Trust (PST), 2011 Ninth Annual International Conference on, IEEE, 2011, pp. 63–70.

[67] BB Gupta, Ramesh Chandra Joshi, and Manoj Misra, *Distributed denial of service prevention techniques*, arXiv preprint arXiv:1208.3557 (2012).

[68] Martin T Hagan, Howard B Demuth, and Orlando De Jesús, *An introduction to the use of neural networks in control systems*, International Journal of Robust and Nonlinear Control: IFAC-Affiliated Journal 12 (2002), no. 11, 959–985.

[69] Wei He and Yiting Dong, *Adaptive fuzzy neural network control for a constrained robot using impedance learning*, IEEE transactions on neural networks and learning systems 29 (2018), no. 4, 1174–1186.

[70] John D Hedengren, Reza Asgharzadeh Shishavan, Kody M Powell, and Thomas F Edgar, *Nonlinear modeling, estimation and predictive control in apmonitor*, Computers & Chemical Engineering 70 (2014), 133–148.

[71] Joseph L Hellerstein, Yixin Diao, Sujay Parekh, and Dawn M Tilbury, *Feedback control of computing systems*, John Wiley & Sons, 2004.

[72] ———, *Feedback control of computing systems*, John Wiley & Sons, 2004.

[73] Chi-Yao Hong, Srikanth Kandula, Ratul Mahajan, Ming Zhang, Vijay Gill, Mohan Nanduri, and Roger Wattenhofer, *Achieving high utilization with software-driven wan*, ACM SIGCOMM Computer Communication Review, vol. 43, ACM, 2013, pp. 15–26.

[74] Sushant Jain, Alok Kumar, Subhasree Mandal, Joon Ong, Leon Poutievski, Arjun Singh, Subbaiah Venkata, Jim Wanderer, Junlan Zhou, Min Zhu, et al., *B4: Experience with a globally-deployed software defined wan*, ACM SIGCOMM Computer Communication Review, vol. 43, ACM, 2013, pp. 3–14.

[75] Jin-Woo Jung, Viet Quoc Leu, Ton Duc Do, Eun-Kyung Kim, and Han Ho Choi, *Adaptive pid speed control design for permanent magnet synchronous motor drives*, IEEE Transactions on Power Electronics 30 (2015), no. 2, 900–908.

[76] Michael CK Khoo, *Physiological control systems: analysis, simulation, and estimation*, John Wiley & Sons, 2018.

[77] Robert Kuschnig, Ingo Kofler, and Hermann Hellwagner, *An evaluation of tcp-based rate-control algorithms for adaptive internet streaming of h. 264/svc*, Proceedings of the first annual ACM SIGMM conference on Multimedia systems, ACM, 2010, pp. 157–168.

[78] Yongming Li, Shaocheng Tong, Lu Liu, and Gang Feng, *Adaptive output-feedback control design with prescribed performance for switched nonlinear systems*, Automatica 80 (2017), 225–231.

[79] Lennart Ljung, *ljung*, Signal analysis and prediction, Springer, 1998, pp. 163–173.

[80] Lennart Ljung (ed.), *System identification (2nd ed.): Theory for the user*, Prentice Hall PTR, Upper Saddle River, NJ, USA, 1999.

[81] Jelena Mirkovic and Peter Reiher, *A taxonomy of ddos attack and ddos defense mechanisms*, ACM SIGCOMM Computer Communication Review 34 (2004), no. 2, 39–53.

[82] Michael Moore, *Ddos attacks increase by 28 percent in q2 2017*, `https://betanews.com/2017/08/23/ddos-attacks-q2-2017/`, (Accessed on 05/20/2018).

[83] Kumpati S Narendra, *Neural networks for control theory and practice*, Proceedings of the IEEE 84 (1996), no. 10, 1385–1406.

[84] Opeyemi Osanaiye, Kim-Kwang Raymond Choo, and Mqhele Dlodlo, *Distributed denial of service (ddos) resilience in cloud: review and conceptual cloud ddos mitigation framework*, Journal of Network and Computer Applications 67 (2016), 147–165.

[85] Charlie Osborne, *The average ddos attack cost for businesses rises to over $2.5 million — zdnet*, `https://www.zdnet.com/article/the-average-ddos-attack-cost-for-businesses-rises-to-over-2-5m/`, (Accessed on 05/20/2018).

[86] Carly Page, *Skype outage to blame on 'ddos attack', hacking group claims responsibility*, https://www.theinquirer.net/inquirer/news/3012275/skype-outage-ddos-attack, (Accessed on 05/20/2018).

[87] Sujay Parekh, Neha Gandhi, Joseph Hellerstein, Dawn Tilbury, T Jayram, and Joe Bigus, *Using control theory to achieve service level objectives in performance management*, Integrated Network Management Proceedings, 2001 IEEE/IFIP International Symposium on, IEEE, 2001, pp. 841–854.

[88] Junhan Park, Keisuke Iwai, Hidema Tanaka, and Takakazu Kurokawa, *Analysis of slow read dos attack*, Information Theory and its Applications (ISITA), 2014 International Symposium on, IEEE, 2014, pp. 60–64.

[89] Tharindu Patikirikorala, Liuping Wang, Alan Colman, and Jun Han, *Hammerstein–wiener nonlinear model based predictive control for relative qos performance and resource management of software systems*, Control Engineering Practice 20 (2012), no. 1, 49–61.

[90] Ragunathan Rajkumar, Insup Lee, Lui Sha, and John Stankovic, *Cyber-physical systems: the next computing revolution*, Design Automation Conference (DAC), 2010 47th ACM/IEEE, IEEE, 2010, pp. 731–736.

[91] Subramani rao Sridhar rao, *Denial of service attacks and mitigation techniques: Real time implementation with detailed analysis*, White paper (2011).

[92] Anders Robertsson, Björn Wittenmark, Maria Kihl, and Mikael Andersson, *Design and evaluation of load control in web server systems*, American Control Conference, 2004. Proceedings of the 2004, vol. 3, IEEE, 2004, pp. 1980–1985.

[93] Martin Roesch et al., *Snort: Lightweight intrusion detection for networks.*, 1999.

[94] Alan Saied, Richard E. Overill, and Tomasz Radzik, *Detection of known and unknown ddos attacks using artificial neural networks*, Neurocomputing 172 (2016), 385 – 393.

[95] Penamakuri Sesha Saikrishna, Ramkrishna Pasumarthy, and Nirav P Bhatt, *Identification and multivariable gain-scheduling control for cloud computing systems*, IEEE Transactions on Control Systems Technology 25 (2017), no. 3, 792–807.

[96] Dale E Seborg, Duncan A Mellichamp, Thomas F Edgar, and Francis J Doyle III, *Process dynamics and control*, John Wiley & Sons, 2010.

[97] D Senie and P Ferguson, *Network ingress filtering: Defeating denial of service attacks which employ ip source address spoofing*, Network (1998).

[98] Johan Sigholm, *Non-state actors in cyberspace operations*, Journal of Military Studies 4 (2013), no. 1, 1–37.

[99] Vasilios A Siris and Fotini Papagalou, *Application of anomaly detection algorithms for detecting syn flooding attacks*, Computer communications 29 (2006), no. 9, 1433–1442.

[100] Qilu Sun, Guanzhong Dai, and Wenping Pan, *Lpv model and its application in web server performance control*, Computer Science and Software Engineering, 2008 International Conference on, vol. 3, IEEE, 2008, pp. 486–489.

[101] Kok K Tan, Qing-Guo Wang, and Chang C Hang, *Advances in pid control*, Springer Science & Business Media, 2012.

[102] Catherine A Theohary and John W Rollins, *Cyberwarfare and cyberterrorism: In brief*, 2015.

[103] Guibin Tian, Yong Liu, Guibin Tian, and Yong Liu, *Towards agile and smooth video adaptation in http adaptive streaming*, IEEE/ACM Transactions on Networking (TON) 24 (2016), no. 4, 2386–2399.

[104] Shaocheng Tong, Shuai Sui, and Yongming Li, *Fuzzy adaptive output feedback control of mimo nonlinear systems with partial tracking errors constrained*, IEEE Transactions on Fuzzy Systems 23 (2015), no. 4, 729–742.

[105] Mustafa Uysal, Pradeep Padala, Xiaoyun Zhu, Zhikui Wang, Sharad Singhal, Arif A Merchant, and Kenneth Salem, *Dynamic feedback control of resources in computing environments*, July 1 2014, US Patent 8,767,535.

[106] Sundarapandian Vaidyanathan, *Adaptive control of a chemical chaotic reactor*, International Journal of PharmTech Research 8 (2015), no. 3, 377–382.

[107] ———, *Anti-synchronization of brusselator chemical reaction systems via adaptive control*, Int J ChemTech Res 8 (2015), no. 6, 759–768.

[108] Jagannadh Vempati, Mark Thompson, and Ram Dantu, *Feedback control for resiliency in face of an attack*, Proceedings of the 12th Annual Conference on Cyber and Information Security Research, ACM, 2017, p. 17.

[109] Ramon Vilanova and Antonio Visioli, *Pid control in the third millennium*, Springer, 2012.

[110] Yanling Wei, Jianbin Qiu, Hak-Keung Lam, and Ligang Wu, *Approaches to ts fuzzy-affine-model-based reliable output feedback control for nonlinear ito stochastic systems*, IEEE Trans. Fuzzy Syst 25 (2017), no. 3, 569–583.

[111] Adrian Wills, Thomas B Schön, Lennart Ljung, and Brett Ninness, *Identification of hammerstein–wiener models*, Automatica 49 (2013), no. 1, 70–81.

[112] Jianxin Yan, Stephen Early, and Ross Anderson, *The xenoservice-a distributed defeat for distributed denial of service.*

[113] Xiaoqi Yin, Abhishek Jindal, Vyas Sekar, and Bruno Sinopoli, *A control-theoretic approach for dynamic adaptive video streaming over http*, ACM SIGCOMM Computer Communication Review, vol. 45, ACM, 2015, pp. 325–338.

[114] ———, *A control-theoretic approach for dynamic adaptive video streaming over http*, ACM SIGCOMM Computer Communication Review, vol. 45, ACM, 2015, pp. 325–338.

[115] Saman Taghavi Zargar, James Joshi, and David Tipper, *A survey of defense mechanisms against distributed denial of service (ddos) flooding attacks*, IEEE communications surveys & tutorials 15, no. 4, 2046–2069.

[116] Hui Zhang, Yang Shi, and Aryan Saadat Mehr, *Robust pid control for multivariable networked control systems with disturbance/noise attenuation*, International Journal of Robust and Nonlinear Control 22 (2012), no. 2, 183–204.

[117] Peng Zhou, Xiapu Luo, Ang Chen, and Rocky KC Chang, *Stor: Social network based anonymous communication in tor*, arXiv preprint arXiv:1110.5794 (2011).