MODELING AND DETECTION OF CONTENT AND PACKET FLOW ANOMALIES AT ENTERPRISE NETWORK GATEWAY

A Dissertation

by

SHENG-YA LIN

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Approved by:

Chair of Committee,	Jyh-Charn (Steve) Liu
Committee Members,	Rabi N. Mahapatra
	Dezhen Song
	Yu Ding
Head of Department,	Duncan M. (Hank) Walker

May 2013

Major Subject: Computer Engineering

ABSTRACT

This dissertation investigates modeling techniques and computing algorithms for detection of anomalous contents and traffic flows of ingress Internet traffic at an enterprise network gateway. Anomalous contents refer to a large volume of ingress packets whose contents are not wanted by enterprise users, such as unsolicited electronic messages (UNE). UNE are often sent by Botnet farms for network resource exploitation, information stealing, and they incur high costs in bandwidth waste. Many products have been designed to block UNE, but most of them rely on signature database(s) for matching, and they cannot recognize unknown attacks. To address this limitation, in this dissertation I propose a Progressive E-Message Classifier (PEC) to timely classify message patterns that are commonly associated with UNE. On the basis of a scoring and aging engine, a real-time scoreboard keeps track of detected feature instances of the detection features until they are considered either as UNE or normal messages. A mathematical model has been designed to precisely depict system behaviors and then set detection parameters. The PEC performance is widely studied using different parameters based on several experiments.

The objective of anomalous traffic flow detection is to detect selfish Transmission Control Protocol, TCP, flows which do not conform to one of the handful of congestion control protocols in adjusting their packet transmission rates in the face of network congestion. Given that none of the operational parameters in congestion control are carried in the transmitted packets, a gateway can only use packet arrival times to

recover states of end to end congestion control rules, if any. We develop new techniques to estimate round trip time (RTT) using EWMA Lomb-Scargle periodogram, detect change of congestion windows by the CUSUM algorithm, and then finally predict detected congestion flow states using a prioritized decision chain. A high level finite state machine (FSM) takes the predictions as inputs to determine if a TCP flow follows a particular congestion control protocol. Multiple experiments show promising outcomes of classifying flows of different protocols based on the ratio of the aberrant transition count to normal transition count generated by FSM.

DEDICATION

To My Wife and Parents

ACKNOWLEDGEMENTS

First of all, I would like to thank Dr. Jyh-Charn Liu for his guidance, support and patience throughout my graduate study. He has been a constant source of inspiration and help. His suggestions and deep insight on academic domain knowledge contributed greatly to the completion of this dissertation, as well as to my academic achievements.

I am immensely grateful to my committee members: Dr. Rabi N. Mahapatra, Dr. Dezhen Song, and Dr. Yu Ding. They gave me very helpful suggestions on improving the quality of this dissertation. I really appreciate their time spent with me on my research.

I would like to express my appreciation to all of my colleagues, and in particular to Mr. Cheng-Chung Tan and Mr. Christopher Bodden. Mr. Tan helped generate important experiment outcomes, which are presented in parts of this dissertation. Mr. Bodden helped proofread and provided comments to improve the readability of the dissertation which I greatly appreciated. I also thank all of the members of the RTDS Lab at Texas A&M University for their research support during my Ph.D. study.

Finally, thanks to my wife and parents for their encouragement, patience and love. Without their support, I would not have finished my Ph.D. degree.

TABLE OF CONTENTS

Page

LIST OF FIGURES

Page

LIST OF TABLES

Page

CHAPTER I

INTRODUCTION AND OVERVIEW OF TECHNICAL CHALLENGES

The Internet is a globally shared resource; its quality of service relies on voluntary cooperation of users. Non-cooperative access of the Internet can cause serious degradation of service quality to some or whole user communities. This dissertation addresses some of the key issues that affect the operations of an enterprise network, which is defined as a network whose users share certain common goals and the network manager has the authority to care for the wellbeing of users. This dissertation aims to address two issues: (a) how to detect voluminous packet contents sent into the enterprise network, and (b) how to detect selfish users who aggressively consume bandwidth, even in the face of network congestion. Both issues are related to capturing the dynamics of network traffic, where the former is at the level of message contents, and the latter is at packet flows. Both are involved with labeling network connections as being "normal" or "anomalous", as their packets flow through. To attack this problem, it is necessary to explore the complex relationship between the computing complexity of the problem, resource constraints, underlying characteristics of the normal vs. anomalous traffic, and eventually the computing and architectural designs. These key issues are highlighted next.

An enterprise gateway is a converging point of both ingress and egress Internet traffic. It is an ideal location to perform enterprise wide management of Internet traffic, especially the proposed network security solutions for detection of anomalous contents

and anomalous traffic flows. Given that a gateway does not have access to hosts, it does not have access to most operational parameters of Internet traffic. As such, several fundamental challenges need to be addressed for detection of anomalous contents and anomalous traffic flows. These challenges are summarized below.

Four key challenges for real-time sensing and detection of UNE:

(1) Memory size and processing costs:

It takes tens or hundreds times more cycles and memory space to process a packet than to transmit the packet. The situation worsens as the transmission bandwidth continues to grow, yet no significant breakthrough is at sight for processor speed or memory density. Decreasing the computing complexity of the processing algorithm is the first and foremost important requirement for content analysis.

(2) Signature based solutions are ineffective for new UNE:

Signature-based detection approach requires generation of the detection signature extracted from confirmed attacks. The long delay in this model makes traditional detection tools ineffective in capturing new UNE.

(3) Computing complexity of the detection system:

Any clues being used for detection of UNE need to be buffered, until a decision can be made to confirm or reject the presence of UNE. The time and storage complexity of algorithms to update those clues are critical to detection performance.

(4) UNE behavior modeling:

Given a fixed amount of resources, an UNE detection system needs allocate resources to detect current UNE while releasing resources allocated to handling of stale UNE. The goal is to maximize the detection rate given an acceptable false alarm.

Major challenges for the detection of non-cooperative packet flows are summarized as follows:

(1) Determining the behavior that can represent different flows:

The flow behavior can be represented by various forms. It is important to define effective features to distinguish well-defined and anomalous TCP flows.

(2) Detecting non-cooperative flows in transient states:

A steady state based modeling technique cannot timely identify anomalous flows. In addition, parameters of these models cannot be easily measured.

(3) Low/high pass filters have limited effects:

Behavior states have different characteristics. Some change slowly, while others change rapidly. When a large number of these hidden behavioral processes are interleaved at a gateway the decision process of detecting non-cooperative flows based on classical filters can become intractable.

(4) Rapid flow parameter changes with noisy background traffic:

Parameters of TCP flows such as RTT and Cwnd change dynamically in response to the level of network congestion. How to determine these parameters that are not explicit parts of packets is very challenging. The challenge is further escalated with loss and retransmission of packets.

All of these problems listed above have not been fully solved in the literature. As it will become clear later in the rest of this dissertation, none of these problems can be effectively solved by straightforward, intuitive techniques. Rather, sound system modeling and efficient algorithms are required to characterize their patterns, and on-line algorithms to capture them.

CHAPTER II

DETECTION OF CONTENT ANOMALY

2.1 Background

For enterprise networks, the ingress Network Access Point (NAP) is a concentration point of unsolicited electronic messages (UNE) before they are dispersed to users. A major advantage of detecting UNE at the NAP is that it provides the largest number of UNE samples at a single point. Real-time filtering of the UNE must avoid adding significant delay to delivery of regular messages. False positive alarms should be minimized to avoid dropping legitimate messages. The memory and computing time requirement need to be bounded in order to handle the large volume of messages.

The anomalous contents not only consume enterprise resources but also jeopardize personal data. Rapid, frequent changes of message contents are a highly effective way for attackers to evade security software, regardless of their sophisticated designs based on information theory, machine learning, and statistics. Driven by profit making, anomalous message distributors continue to launch enormous amounts of malicious messages that flood enterprise networks, consuming network resources and posing a severe threat to the credibility of legitimate message communications. Taking a hit and run, highly mobile message delivery and (victim) harvesting strategy, educated anomalous message distributors routinely defeat message filters that rely on trained signatures. We argue that a critical missing link of existing network security management tools is their ability in real-time sensing and classifying of UNE signatures,

so that the countermeasures can be deployed proactively before the attack spreads too widely to be manageable.

In this dissertation, we propose a real-time anomaly detector called Progressive Electronic-Message Classifier (PEC) [1] for early detection of UNE before they spread into the local networks. To meet the performance challenges, an O(1) spinning wheel algorithm is developed for the aging-scoring function. In addition, a mathematical model is created to derive system parameters such as the score threshold and age table size given the specific detection latency and successful probability.

PEC detects UNE based on their self-similarity property, which refers to an unusually large number of identical/similar message structures and/or contents such as the URL links, message structures, or the last few SMTP relay hops, etc. that can effectively identify the onset of a wave of UNE that were freshly crafted in a new format and sent from some unknown sources. PEC can be used as a behavior based filter alone, or as needed, it can be interfaced to existing anti-spamming tools so that message samples that trigger the anomaly detectors could be further examined.

Unlike most commercial spamming filters based on statistics or pattern training, PEC detects surging of feature instances in predefined features without prior training. A feature can be any data type, e.g., html statements like URL links or IMG sources, sender sources, etc., that represent invariant characteristics of UNE. Message components not defined by the detection features are considered variants of the UNE that would be ignored by PEC. On the basis of an O(1) spinning wheel algorithm to keep

track of feature instances, the core detection engine of PEC can also be used in other protocol layers for real-time anomaly detection without prior training.

An experimental prototype of PEC integrated to sendmail is implemented for detection of URL links of UNE, albeit the system can be easily expanded to filter other features. URL links extracted from the corpus of spamming-phishing messages are used to generate UNE, which are then mixed with regular messages to test the prototype. To meet the performance challenges, an O(1) spinning wheel algorithm is developed for the aging-scoring function. Experimental results on a Xeon based server show that PEC can handle 1.2M score updates, hashing and matching of 7k URL links, and parsing of 200 messages bodies of average size 1.5kB. The lossy detection system can be easily scaled up and down by progressive selection of detection features and detection thresholds. It can be used alone or as an early screening tool for existing infrastructure to defeat major UNE flooding.

2.2 Existing Solutions

Significant progress has been made in the areas of host or local message server protection: virus protection (McAfee, Trend, Symantec), host intrusion detection systems [2][3], system quarantine efforts, user-agent spam filtering, and even policy requirements for system matches **[**4**]**. Admittedly, there are a number of techniques designed to detect UNE, but the adversary has a broad spectrum of tricks and means to deceive. The adversary is always a step ahead.

Some of the most noted systems include intrusion detection systems (IDS), antivirus and anti-spammer software systems that rely on string matching and parsing to capture known attack patterns. Popular IDS software, such as L7-filter [5], Snort, and Bro, parse and match patterns (attack vectors) as a component of their finite state machines (DFA or NFA) to represent patterns. Thus, they require accelerated parsing, an active research area, e.g., grouping of patterns [6], and multiple regular expressions matching [7]. The exponential growth in memory space consumption in these classical algorithms suggests that they are more suitable for detailed inspection of messages. Commercial tools can be used sparsely for detailed analysis of a small number of messages, but the computing cost is too high to use them for inspection of every message at wire speed. We will focus on algorithm aspects for an overall system solution in the current study. Hardware solutions [8][9] will not be considered.

A myriad of solutions have been crafted to filter messages. Approaches range from a simple policy to only accept messages from known senders, to those based on Bayesian inference based on statistical key word matching. Black/white lists are based on sharing of the databases of known good and bad sources. The computationally feasible puzzles technique developed decades ago is being revisited as a rate limiting technique to deter spam [10]. Black lists (SpamCop, Sorbs, etc.) of spamming sources and phishing web sites are being tracked by organizations [11][12]. Among other measures, global message providers adopt source authentication. Gmail and Yahoo both adopted the DomainKeys technology, which aims to detect spoofing senders by using

encrypted tags to be compared with a shared or public database of active spamming Internet addresses [13].

Spamassassin [14] filters spamming messages based on a compound scoring system calculated from Bayesian statistics, DNS and URL black lists, and various spam databases. Implemented in Perl, [15] it is not designed for line-speed inspection. PILFER [16] uses ten features e.g., IP-based URLs, non-matching URLs, JavaScript, etc. to detect phishing frauds. The features were generated using off-line SVM. Spamato [17] is another popular spam filter, which provides an extendable interface to support plug-ins of third-party filters and makes score decisions using the filter outcomes. Currently it supports Vipul's Razor Filter [18] (blacklisted domain name look up), Earl Grey Filter [19], etc. Dominator [20] is also a URL-based filter, which uses "URL+spam" or "URL+blacklist" as the keyword to inquire Google for the likelihood of spamming. SpamGuru [21] supports user voting through Lotus Notes mail clients to handle black/white lists. A pipelined-based architecture allows multiple filers cascaded together to perform analysis of a message until it can be declared a spam or otherwise.

The sophisticated computations in content analysis based filters limit them to non real-time environments. On the other hand, PEC is designed as a line-speed detector to capture surges of new feature instances in messages. It should not come as a surprise to see integration of the two classes of detectors in different ways. For instance, PEC can use existing tools to determine the likelihood of spamming by a newly detected surge of a feature instances at the ingress of the enterprise network. Or, PEC can publish the

score results so that existing spam filters can use the published list to adjust their score weights at servers or desktops.

Behavior profiling is a viable technique to differentiate normal vs. anomalous patterns of messages [22] and other general usage [23][24][25] based on usage frequency, social cliques, and interactions. Selection of the distance function needs to be done based on the nature of detection features [26]. The n-gram analysis technique has been used to detect malicious code execution and anomalous message generation patterns. Behavior patterns could be measured by distance functions. They can range from simple bit by bit comparison (Hamming distance) to information theoretic measures (e.g., Kullback-Leibler test). A general distance function is $\|\mathbf{x}-\mathbf{y}\|_{\mathbf{p/r}} = \left(\sum_{\mathbf{i=1}}^{\mathbf{D}} |\mathbf{x_i}-\mathbf{y_i}|^{\mathbf{p}}\right)^{\mathbf{1/r}}$ [27], where x and y are feature vectors, p-weight on dimension dissimilarity, and r the distance growth of patterns. Selection of the distance function needs to be done based on the nature of detection features (numerical vs. string, point observations vs. multivariate inputs, time window size and sampling methods, etc.) The ultimate judgment on selection of detection features and their distance functions needs to be based on their detection performance on the real data.

Spamming messages can also be viewed as a type of heavy hitters of network traffic. However, despite the similar nature between PEC and these network heavy hitter detectors, they operate on entirely different premises because PEC is a point detector but its counterparts are for Internet wide analysis. Hierarchical aggregation and computation of multi-dimensional data (IP address, port number, and protocol fields of IP flows, etc) have been extensively investigated, e.g., [28][29], where a heavy hitter is essentially

traffic flow whose volume, which is computed from the hierarchical data structure, exceeds a certain threshold. Once detected, one could even locate the source via a high dimensional key space. A deterministic sampling technique called lossy counting (LC) [30] was proposed to detect heavy hitters with bounded errors. The sketch-based approach [31] used a small amount of memory to detect anomalous traffic. Based on it, a reverse hash method [32] is used to identify the keys of culprit flows without extra recording memory.

Intrusion detection systems (IDS) and antivirus software systems also rely on string matching and parsing to capture known attack patterns. Popular IDS software, such as L7-filter [5], Snort [1], and Bro [3], parse and match patterns (attack vectors) as a component of their finite state machines (DFA or NFA) to represent patterns. Hardware and software based parsing acceleration [6][7][8][9][27] is a highly active research area.

Yet, there is no solution to address UNE at the entrance points of an enterprise network. It is likely that these approaches and other statistical similarity functions, in their original forms, would have limited use for on-line detection because of their high computing complexity. Thus, the motivation behind this research is to devise a new form of computer network defense to protect military networks from threats posed by seemingly innocent UNE. We will focus on fast algorithms tailored for detecting the onset of structural or content similarity (word count and number of URL links, image rendering sources, etc) of buffered messages in a time window, so that the content

similarity can be extracted by a higher-level of analysis and can configure UNE detectors on the fly.

2.3 PEC System Architecture

PEC is designed to detect anomalous surges of feature instances (FI) of major UNE flooding. An FI is a particular realization of the UNE feature F, which is any message construct that is likely to be used by spammers. From the viewpoint of system event management, each detected FI represents one feature clock (FC) that drives all state updates. To operate in a broadband environment, the detection process in PEC is made dynamic and lossy, i.e., some messages may pass the (overloaded) detector without being checked in order to avoid adverse performance effects due to any type of resource contention.

Let F = { α1, α2, α3 …, } represent a set of binary strings which can be expressed and parsed by a finite automata, and αi of F is an FI of F. A message construct is not a viable UNE feature if it cannot be effectively used to discriminate regular messages from UNE, e.g., the greeting words, subject line, etc. As of the writing of this dissertation, key words/phrases (“click here”, “getting rich”), deceptive URL links, or remotely rendered images are the most prevalent UNE features. The detection features can be added/removed (“progressive”) based on resource constraints and user requirements.

Let γ denote a newly identified FI by PEC, γ is assigned one of three states: Xγ ← G/B/W, i.e., Gray (unchecked), Black (UNE), or White (not UNE), until it is removed

from the system. Xγ (v)← G, where v is the current FC value. γ will be retained for a certain time period before its state changes, i.e., ~~Xγ~~W/B. Xγ ← B (from G) if the number of its occurrences, called score, Rγ exceeds a score threshold, S, but Xγ← W if its age, Aγ, exceeds an age threshold, M, the age of γ is the time elapsed before its score is increased. S and M are two major design parameters that decide the detection sensitivity and false alarm rates of the system.

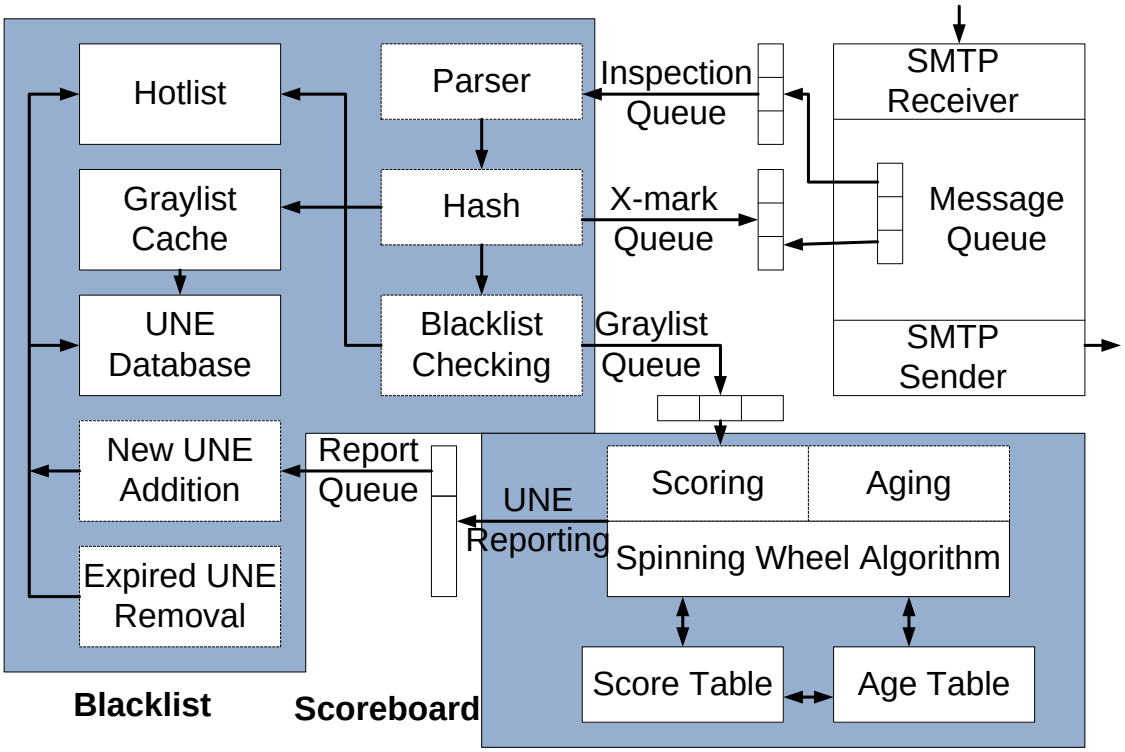


Figure 1: PEC architecture.

Referring to Figure 1, we propose a cascaded filter architecture consisting of a blacklist and scoreboard to track FIs. Messages being filtered are parsed for FIs by the feature parser(s) in the blacklist module. After an FI γ is extracted and hashed to Hγ, Hγ is checked against the hotlist, which is the hash vector of currently active FIs in some

UNE. If a hit occurs to Hγ on the hotlist, it means that γ is an UNE instance, and the SMTP server could take countermeasure action, e.g., X-mark the (UNE) message through the X-mark queue, or simply increase its hit count. Otherwise, (γ, Hγ) is placed on the graylist cache, and Hγ is placed on the graylist queue of the scoreboard for further tracking.

The hotlist is a single bit array with all or a part of Hγ as its table address. The graylist cache serves as a temporary lookup table between FI and HFI. (γ, Hγ) is moved into the UNE database once it appears on the report queue, i.e., Rγ was found to exceed S by the scoreboard. Or, (γ, Hγ) is simply removed from the cache when Aγ exceeds M. The average life span of UNE feature instances is short. As such, a background thread periodically examines the UNE database so that the FI can be removed when it becomes cold.

Hγ passed from the blacklist to the scoreboard is tracked by scoring and aging functions based on a competitive aging-scoring scheme (CASS). Each FI that does not hit the blacklist but enters the scoreboard generates a virtual clock (VC) which triggers all of the state transitions. In the rest of the discussions we will use v to denote the current VC value, unless explicitly defined otherwise.

If Hγ is new, it is placed into the score table (ST) and its score Rγ ← 1, and age Aγ ← 0 (in the age table (AT)). Otherwise, if Hγ is already in ST, Rγ is incremented and Aγ ← 0. Three types of operations are interlocked in each VC: increase of Rγ, rest of Aγ, and increase of Aβ, β≠ γ. An FI that does not have its score increased for a

consecutive number of VCs is eliminated from the ST, i.e., $X_{FI} \leftarrow W$ and FI is not an UNE feature. CASS is summarized as follows.

R1. If $H\gamma$ (or equivalently γ), is already in the scoreboard, then $R\gamma$ increments, and $A\gamma \leftarrow 0$. Otherwise, insert it into the ST with $R\gamma \leftarrow 1$, and also into the AT with $A\gamma \leftarrow 0$. An old entry $H\tau$ in ST and AT may need to be removed from the scoreboard, i.e. $X\tau \leftarrow W$, if $A\gamma > M$ happens to occur at the same VC.

R2. $A\beta$ increments and $R\beta$ unchanged, where $\beta \neq \gamma$,

R3. Report $H\gamma$ to blacklist if $R\gamma > S$; remove $H\gamma$ from scoreboard after reporting.

A critical design goal of CASS is to reduce the O(N) computing cost in age increment of a naïve implementation to O(1). To solve this problem, we propose a spinning wheel algorithm to keep track of ages of entries by modeling AT into a cyclic queue, and using the queue location of an entry to represent its age. This way, the scoring (of γ) and aging operations, i.e., steps R1-R3, of all entries in AT can be collapsed into a few fixed computing cycles, independent of the sizes of AT and ST.

Different hash functions can be used for H. For simplicity and computing speed, we adopt the SDBM hash function [33] for PEC. Hash collision is inevitable, but this problem can be remedied by adding a further step of exact string matching or additional hashing. To keep our current discussion focused on the system architecture, we will address the hash management issue in the future.

2.4 Low Computing Cost Algorithm

2.4.1 Scoring Function

Similar to the hotlist, the score table (ST) could use the hash values of FIs as its address directly when a large number of FIs need to be tracked. Given that FIs would only stay in the scoreboard for a relatively short time before they are classified as UNE (B) or normal (W), we propose to only use a fraction of the hash value to implement ST based on a pointer table and linked lists, such as the data structure illustrated in Figure 2. Here, each linked node consists of a score, AT-index, and stamp. The score field keeps track of Rγ. The AT-index field points to the AT location of Hγ in the AT, L, and ATL ← Hγ if Hγ is a new FI so that a reverse look up can be made from any AT location of its associated linked node through the pointer table. The stamp is derived from a fraction of Hγ bits as follows.

When an entry Hγ in the graylist queue is being processed by the scoreboard, Hγ is divided into two parts, $\mathbf{H}_{\boldsymbol{\gamma}}^{\mathbf{A}}$ and $\mathbf{H}_{\boldsymbol{\gamma}}^{\mathbf{C}}$, respectively, where $\mathbf{H}_{\boldsymbol{\gamma}}^{\mathbf{A}}$ serves as the address of a pointer table entry, which points to a linked node N, and then $\mathbf{H}_{\boldsymbol{\gamma}}^{\mathbf{C}}$, is checked against the stamp field of N. If a hit in ST occurs to Hγ, its score Rγ is incremented, and if Rγ becomes larger than S, Hγ is sent to the report queue (to the blacklist), and its entries in both AT and ST are nullified. All nodes linked to N may need to be checked until a hit occurs, or a new node will be created for Hγ. By passing L to the aging function, we can adjust the ages of all entries in the AT. We note that any of the k bits in Hγ can be assigned to $\mathbf{H}_{\boldsymbol{\gamma}}^{\mathbf{A}}$, but experiments show that using the lowest order bits tends to have lower collision ratios (or shallower linked lists).

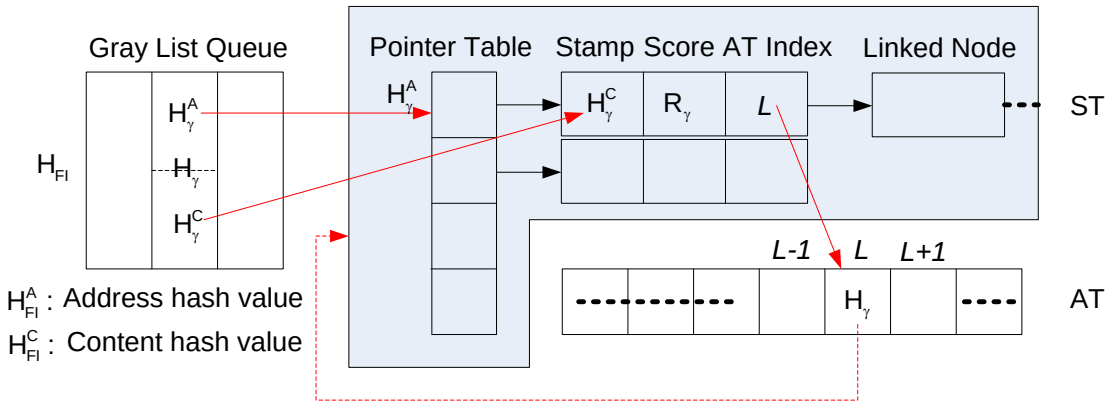


Figure 2: Score table and age table.

2.4.2 Aging Function

The aging function manages the ages of FIs in the scoreboard by modeling AT into a cyclic queue (CQ), where the CQ position of Hγ represents its relative age among all entries in the scoreboard: Let the head and tail of CQ be denoted as $\mathbf{CQ_D}$ and $\mathbf{CQ_T}$, respectively. The age of Hγ is smaller than that of Hβ if it is positioned closer to $\mathbf{CQ_D}$ than $\mathbf{H_\beta}$ is. For a CQ entry placed at the $\mathbf{AT_i}$, its (next) adjacent queue entry is located at $\mathbf{AT_{(i+1 \bmod M)}}$, where M is the table size, and $\mathbf{AT_0}$ is the first array location.

Recall that scoring of one FI also implies aging of all other entries. To age all other FIs in the scoreboard at the lowest cost, first we make the age threshold M, also the size of AT, and we move $\mathbf{CQ_D}$ from its current location in AT to that of $\mathbf{CQ_T}$ at each VC. The notion of "spinning wheel" is attributed to shifting of the new $\mathbf{CQ_D}$ to that of the existing $\mathbf{CQ_T}$, as illustrated in Figure 3.

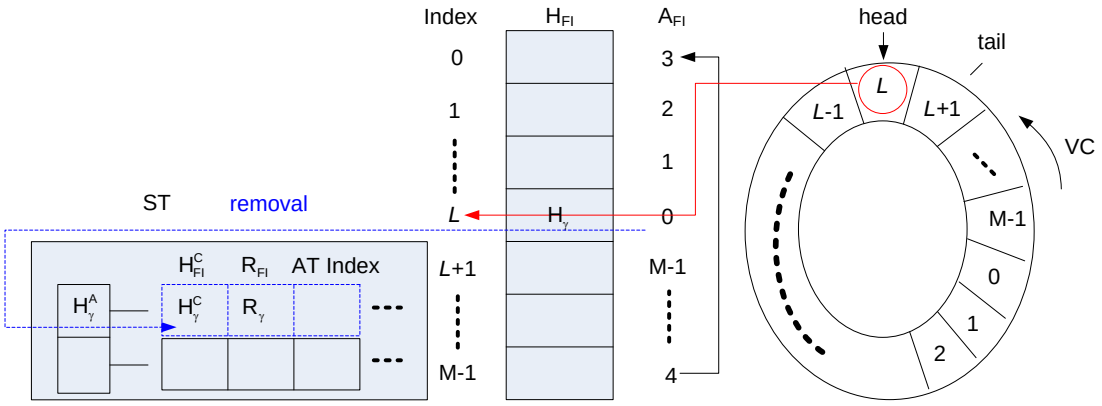


Figure 3: The spinning wheel algorithm in a CQ.

The entry Hγ retrieved from the graylist queue into the scoreboard is placed at the new $\mathbf{CQ_D}$ position (in addition to proper ST updates) because it is made the "youngest" entry in scoreboard by CASS rules. Now that M is both the size of AT and the age threshold, a non-null $\mathbf{H_\beta}$, $\mathbf{H_\beta \neq H_\gamma}$, located at the (existing) $\mathbf{CQ_T}$ is the only entry whose age is to exceed M. As a result, these few actions required to realize R1, R2 and R3 can be computed in a few fixed steps, regardless of the value of M. Of course, entries of ST and AT for both $\mathbf{H_\beta}$ and $\mathbf{H_\gamma}$ need to be updated to maintain the overall consistency. It is trivial to prove that the computing complexity of the spinning algorithm is O(1).

In summary, AT entries need to be adjusted in the current VC after the aging function receives a request from the scoring function to handle Hγ based on three cases: (1) Rγ = 1, (2) 1< Rγ≤S, (3) Rγ = S+1. In case (1), γ is a brand new FI and therefore , shift the AT location of the new CQ_D to that of the existing CQ_T. Then write Hγ into the new CQD location. In case (2), γ is already in the scoreboard, but Rγ is not high enough

to be considered UNE. Four steps are needed for this case: Step (a) use the AT-index of $\mathbf{H_\gamma}$ in ST to locate the AT table address of $\mathbf{H_\gamma}$ and nullify its content. Step (b) change the AT location of the new $\mathbf{CQ_D}$ to that of the existing $\mathbf{CQ_T}$. Step (c) copy $\mathbf{H_\gamma}$ into the new $\mathbf{CQ_D}$ location. Step (d) copy the new $\mathbf{CQ_D}$ location back to the AT-index field of the $\mathbf{H_\gamma}$ node in the ST. In case (3), $\mathbf{R_\gamma}$ exceeds the score threshold and needs to be passed to the blacklist. We nullify the AT entry of $\mathbf{H_\gamma}$, (erase the $\mathbf{H_\gamma}$ node in ST), and shift and move the location of the new $\mathbf{CQ_D}$ to that of the existing $\mathbf{CQ_T}$ as usual. Of course, any non-null entry $\mathbf{H_\beta}$ in the existing $\mathbf{CQ_T}$ needs to be purged from the scoreboard.

The interlocked operations between the scoring and aging functions are illustrated through **Figure 4** consisting of three FIs: α, β and γ. The states of each FI are tracked by its score and age. Its state (W/G/B) is marked by very light, gray and solid black horizontal line segments. The event that triggers the state of an FI to change is marked by the arrowed curves. The arrival of an FI represents one VC tick. For ease of presentation, we draw the $\mathbf{R_{FI}}$ and $\mathbf{A_{FI}}$ in positive and negative Y directions, respectively. Note that VCs are not uniformly distributed on the real time line in this example.

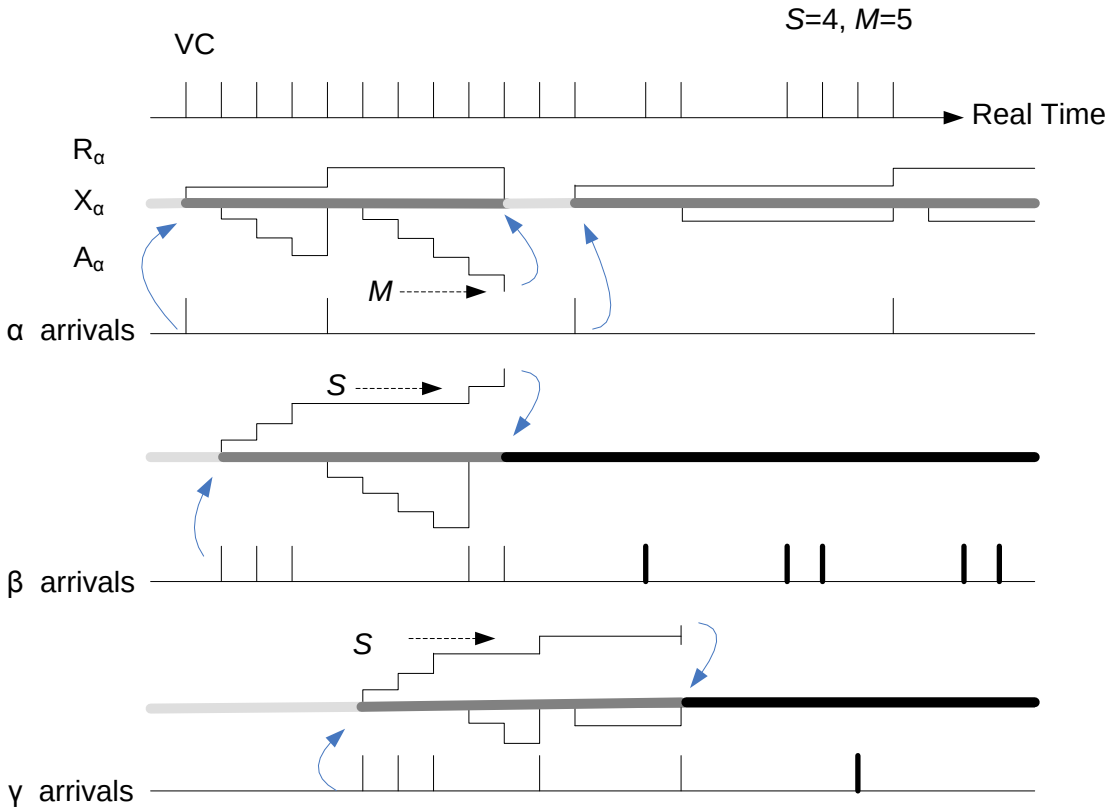


Figure 4: An illustration of aging-scoring processes in CASS.

2.5 Modeling of the PEC Behavior

Any FI will stay in the scoreboard for a finite amount of time before it is purged with its state labeled as B/W. The sojourn time for a normal FI is simply M counts of VCs, but it is not as obvious in determining that of an UNE FI, i.e., the detection latency. Instead of full assessment of detection, sensitivity and selectivity in the classical sense which would require extensive field experiments, we propose to investigate the relationship between the detection latency (time elapsed before UNE are placed in the

blacklist) and the two key parameters M and S. That is, our goal is to answer the following two questions.

Q1: "What is the minimum value of M to detect an UNE attack (of known density) with a success probability of higher than α?"

Q2: "For a given M, what is the maximum value of S to guarantee that a probability measure P(detection latency < ς) > α ?"

Scoring and aging of FIs in the CASS constitute a competition process among FIs in the scoreboard. To answer the aforementioned questions, we only consider a single UNE source condition here. It will become clear from simulations that this represents a worse case scenario than multi-UNE source cases when input rates are fixed.

Answer to Q1:

Let foreground and background events, $\mathbf{E_f}$ and $\mathbf{E_b}$ denote UNE and normal traffic, whose average rates are $\boldsymbol{\mu_f}$ and $\boldsymbol{\mu_b}$, respectively. Without loss of generality, we assume that $\boldsymbol{\mu_b} > \boldsymbol{\mu_f}$. Recall that the arrival of each FI represents a VC tick; let λ denote the mean rate measured by instances of $\mathbf{E_f/VC}$, then

$$\boldsymbol{\lambda} = \frac{\boldsymbol{\mu_f}}{\boldsymbol{\mu_b} + \boldsymbol{\mu_f}}. \quad (1)$$

Given λ , we are interested in determining the relationship between M and α, the probability that the instance of $\mathbf{E_f}$ occurs more than once in M VCs. Consider the more challenging case of low density UNE detection, i.e., $\boldsymbol{\mu_f} \ll \boldsymbol{\mu_b}$, λ can be approximated as the rate of a Poisson process $\{\mathbf{N_f(\tau)}, \boldsymbol{\tau} \geq \mathbf{0}\}$, where $\mathbf{N_f(\tau)}$ is is the number of $\mathbf{E_f}$

happening prior to τ VCs. Since an $\mathbf{E_f}$ instance will be purged from the scoreboard if and only if $\mathbf{N_f(t+M) - N_f(t) \leq 1}$, then

$$\boldsymbol{P(N_f(t+M) - N_f(t) \leq 1)} \quad (2)$$

is the probability of an $\boldsymbol{E_f}$ instance expiring in M VCs, i.e., where P is the probability density function of the Poisson distribution [34], expressed by

$$\boldsymbol{P\big(N_f(t+M) - N_f(t) = n\big) = e^{-\lambda t}(\lambda \Delta t)^n / n!}. \quad (3)$$

In addition, the cumulative density function, CDF, of the Poisson distribution is

$$\boldsymbol{P\big(N_f(t+M) - N_f(t) = n\big) = \Gamma(n+1, \lambda \Delta t)/n!} \quad (4)$$

where $\boldsymbol{\Gamma}$ is the incomplete gamma function defined as $\boldsymbol{\Gamma(x,y) = \int_y^\infty t^{x-1} e^{-t} dt}$. Thereby given the survival probability α of an $\boldsymbol{E_f}$ instance and a known λ, M can be computed using

$$\boldsymbol{\Gamma(2, \lambda M) = 1 - \alpha} \quad (5)$$

Figure 5 shows a pivoting point located at λM = 5 of the Γ mapping curve. It means that the survival probability of an $\boldsymbol{E_f}$ instance only has a marginal increase after the size of AT exceeds 5/λ .

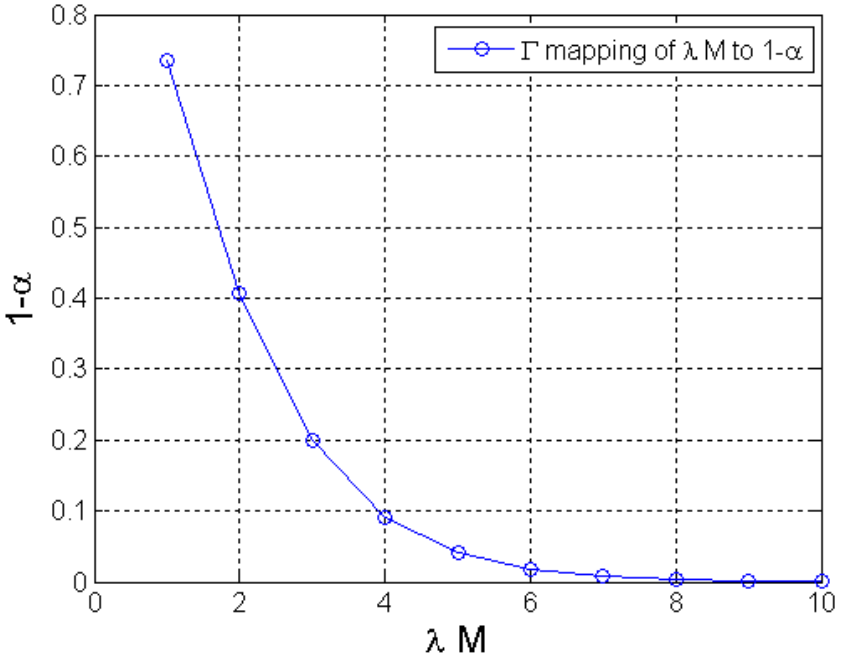


Figure 5: The incomplete Γ mapping between λM and $\mathbf{1-\alpha}$.

Answer to Q2:

Next, we discuss the setting of S in Q2. First, we consider the expected number of $\mathbf{E_f}$ instances before it is declared UNE for a given probability value α and S. Let $\mathbf{H_k}$ denote the number of hits of $\mathbf{E_f}$ until k consecutive effective hits (CEH) occurs, where an effective hit of $\mathbf{E_f}$ denotes the scenario that $\Delta t \leq M$, when a new $\mathbf{E_f}$ instance occurs, where Δt represents the time interval between two adjacent $\mathbf{E_f}$ instances. That is, the score of $\mathbf{E_f}$ does increase for this hit, and it is not kicked out due to aging. If it takes $\mathbf{H_{s-1}}$ VCs to obtain S-1 CEH, then either the next time is an effective hit and we have S CEH, or it is not an effective hit and the scoring procedure of the $\mathbf{E_f}$ instance must begin

anew. For an $\mathbf{E_f}$ instance, its effective hit probability is α, otherwise, 1−α. The expected value of $\mathbf{H_s}$ can be expressed as

$$\mathbf{E}(\boldsymbol{H_s}|\boldsymbol{H_{s-1}}) = \boldsymbol{\alpha}(\boldsymbol{H_{s-1}} + \mathbf{1}) + (\mathbf{1} - \boldsymbol{\alpha})(\boldsymbol{H_{s-1}} + \mathbf{1} + \mathbf{E}[\boldsymbol{H_s}]) \tag{6}$$

i.e.

$$\mathbf{E}(\boldsymbol{H_s}|\boldsymbol{H_{s-1}}) = \boldsymbol{H_{s-1}} + \mathbf{1} + (\mathbf{1} - \boldsymbol{\alpha})\mathbf{E}[\boldsymbol{H_s}] \tag{7}$$

Taking expectations on both sides of the preceding yields

$$\mathbf{E}(\boldsymbol{H_s}) = \boldsymbol{E}(\boldsymbol{H_{s-1}})/\boldsymbol{\alpha} + \mathbf{1}/\boldsymbol{\alpha} \tag{8}$$

Since H1, the number of $\boldsymbol{E_f}$ until the first effective hit, is equal to one, we see that $\mathbf{E}(\boldsymbol{H_1})$=1, and recursively

$$\begin{aligned} \boldsymbol{E}(\boldsymbol{H_2}) &= \frac{\mathbf{2}}{\boldsymbol{\alpha}}, \\ \mathbf{E}(\boldsymbol{H_3}) &= \frac{\mathbf{1}}{\boldsymbol{\alpha}} + \frac{\mathbf{2}}{\boldsymbol{\alpha^2}}, \\ \mathbf{E}(\boldsymbol{H_4}) &= \frac{\mathbf{1}}{\boldsymbol{\alpha}} + \frac{\mathbf{1}}{\boldsymbol{\alpha^2}} + \frac{\mathbf{2}}{\boldsymbol{\alpha^3}}, \end{aligned} \tag{9}$$

and in general,

$$\begin{aligned} \mathbf{E}(\boldsymbol{H_s}) &= \frac{\mathbf{1}}{\boldsymbol{\alpha}} + \frac{\mathbf{1}}{\boldsymbol{\alpha^2}} + \frac{\mathbf{1}}{\boldsymbol{\alpha^3}} + \cdots + \frac{\mathbf{1}}{\boldsymbol{\alpha^{(S-1)}}} + \frac{\mathbf{2}}{\boldsymbol{\alpha^S}} \\ &= (\mathbf{1} - \mathbf{2}\boldsymbol{\alpha^{-S}} + \boldsymbol{\alpha^{-S+1}})/(\boldsymbol{\alpha} - \mathbf{1}) \end{aligned} \tag{10}$$

where $\mathbf{E}(\mathbf{H_{s+1}})$ implies the expected number of $\boldsymbol{E_f}$ instances before it is declared an UNE. With this result, we can determine the detection latency of $\boldsymbol{E_f}$ in terms of physical time. With the probability α, the average detection latency ς expressed in terms of VC is

$$\varsigma = \boldsymbol{E}(\boldsymbol{H}_{s+1})/\boldsymbol{\lambda} \quad (11)$$

Or, when expressed in terms of physical time (seconds),

$$\varsigma = \boldsymbol{E}(\boldsymbol{H}_{s+1})/\boldsymbol{\mu}_{f} \quad (12)$$

Example:

We use the following simple example to demonstrate the setting of M and S values for a target detection rate and latency. Here, the average rate of $\mathbf{E_f}$ is 1.5, and that of $\mathbf{E_b}$ 15.0. Our design goal is to detect $\mathbf{E_f}$ in 520 VCs given the timely detection rate (TR) $\alpha = 0.96$, i.e., the percentage that the actual detection latency is no more than the expected latency.

Since $\mathbf{E_f} = 1.5$ and $\mathbf{E_b} = 15.0$, we can thereby get $\lambda = 0.091$ using Equation (1). M = 55 can be derived by plugging $\lambda = 0.091$ and $\alpha = 0.96$ into Equation (2) such that $\Gamma(2,0.091*M) = 0.04$. To detect the $\mathbf{E_f}$ instance in 520 VCs, it means that we have $\mathbf{E(H_{s+1})} = 47.320$ by using $\varsigma = 520$ in Equation (11). To meet the detection latency goal, a larger S value is preferred over a smaller one, so that the false positive alarms would be lower. By plugging S = 23, 24, and 25 respectively into Equation (10), we can obtain $\mathbf{E(H_{24})} = 44.257$, $\mathbf{E(H_{25})} = 47.143$, and $\mathbf{E(H_{26})} = 50.149$. As a result, we choose S = 24 among the three choices because it is the largest S value which can meet the detection latency requirement.

To evaluate the accuracy of the model, we made three major experiments to examine (i) the actual detection latency using the model-derived parameters, i.e., S = 24 and M = 55 , (ii) the TR by varying S while M = 55, and (iii) the TR by varying M but bounding S to 24. For each major experiment, we have 10 samples, numbered from 1 to

10; each sample is based on 100 runs, and the overall experimental results are plotted in Figure 6.

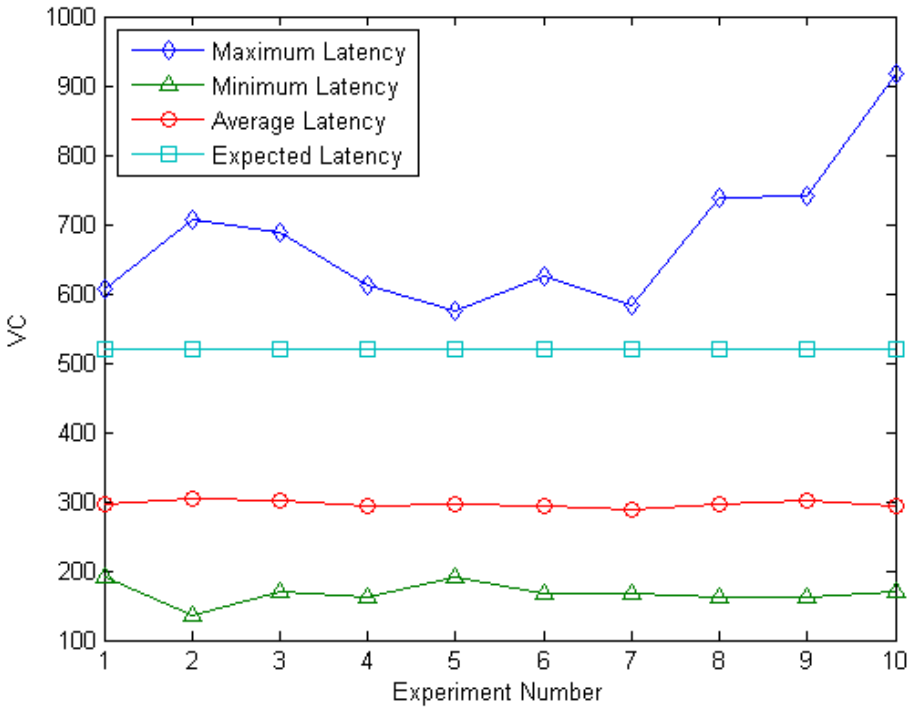


Figure 6: Detection latency based on S=24, M=55.

From Figure 6 it shows that the average detection latency for each sample is around 300 VCs, which is much lower than the model based estimation, 520 VCs. This means that in most cases the $\mathbf{E_f}$ instance can be detected in 520 VCs, but in some cases the $\mathbf{E_f}$ instance is detected in more than 520 VCs, and it is called the tardy detection rate (RR), RR=1-TR.

Next, we measured the effects of S on TR with the change of S, and the results are plotted in Figure 7. Among the different parameter settings, pair (S, M) = (24, 55) gives the best experimental fit to the model based estimation of TR, which is 0.96. When

S is reduced to 22, TR increases to [0.96, 0.99], and when S is increased to 26, the median of its TR becomes 0.94, which is lower than 0.96.

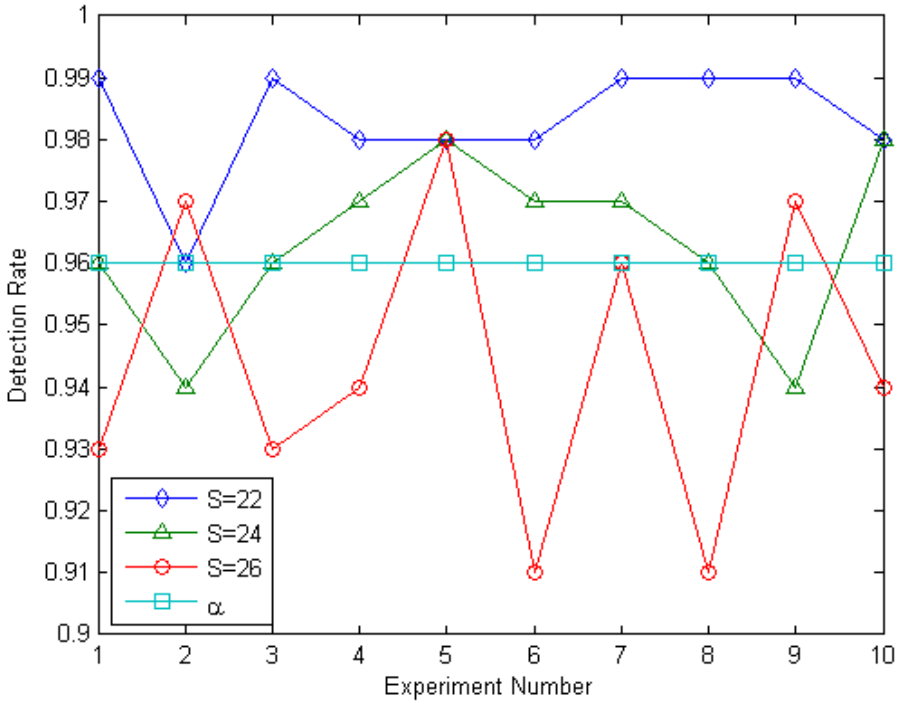


Figure 7: Variation of the detection rate based on M=55

Finally, we measured the effect of M on TR, and the results are plotted in Figure 8. Both the sojourn time of the $\mathbf{E_f}$ and the detection probability increase with M. When M is increased to 57, the mean value of TR (line marked with circle icons) is increased to 0.97. When M is reduced to 53, the mean value of TR (line marked with diamond icons) is decreased to 0.95.

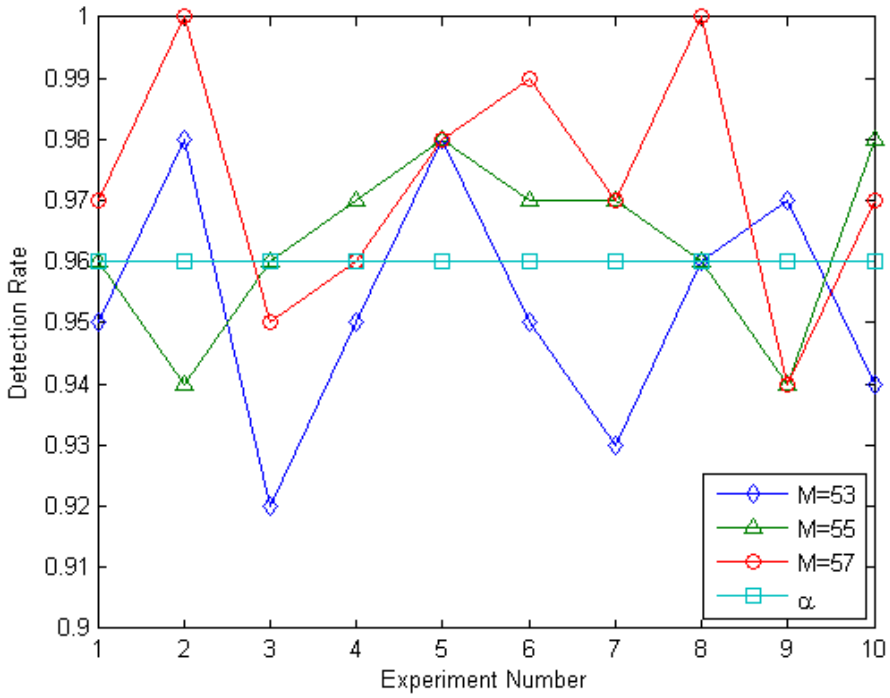


Figure 8: Variation of the detection rate based on S=24.

CHAPTER III

PERFORMANCE OF CONTENT ANOMALY DETECTION

3.1 Experiment Configuration

A prototype of PEC was developed to evaluate its performance in a 100 Mbps local network. The prototype has four elements, as illustrated in Figure 9. PEC and the sendmail daemon run on a Dell PowerEdge 1420 with a Xeon 3.0 GHz CPU and 2GB memory. The control console and message senders run in Windows XP based PCs.

The first prototype element is the message generator together with a SMTP client to generate regular and UNE messages following the experiment instructions sent from the control console. The control console accepts user commands (in XML) to set up experiment parameters. It also accepts user commands to set the sizes of the blacklist, ST, and AT. The control console also has a graphical interface to display the runtime status of PEC, such as the detected UNE, normal messages, utilization ratio of the AT, and the full contents of the detected UNE. The third element is the sendmail 8.14.1 SMTP daemon [35] which runs on a dedicated machine with standard port setup. Sendmail can handle up to 20 messages per second in our experiments.

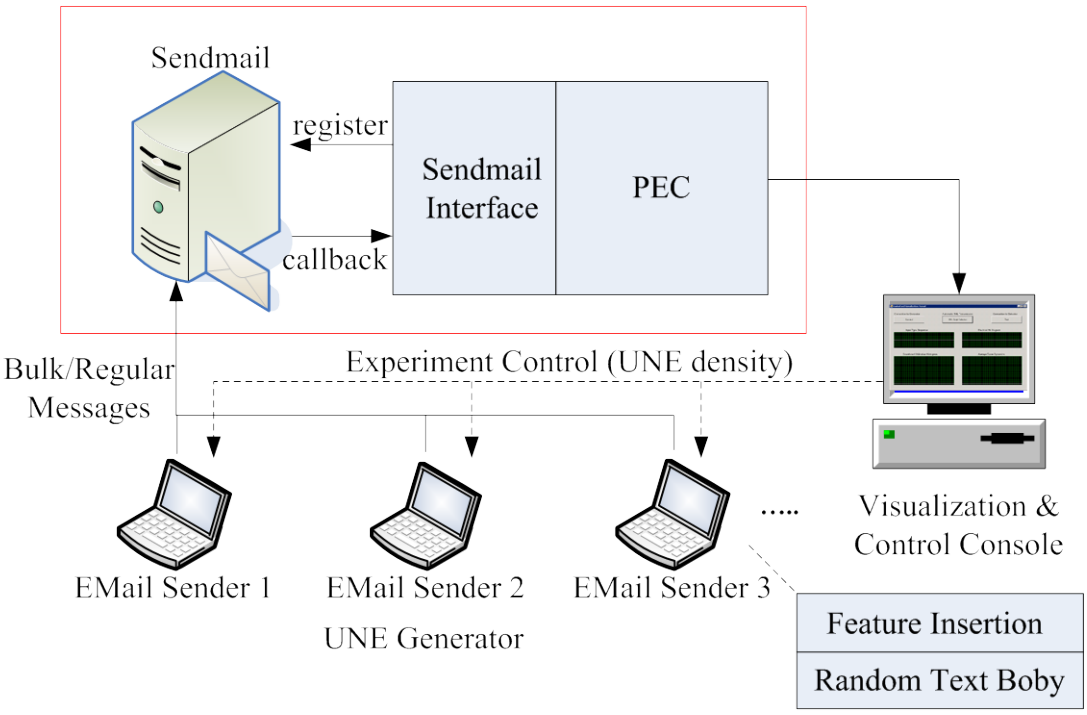


Figure 9: Experimental set up of UNE Detection.

The fourth element is the PEC, which is interfaced to the sendmail daemon through its standard registration process, so that a set of callback functions are exposed to sendmail. Sendmail can also expose basic message structures, i.e., header, message body, end of file, etc. to PEC. PEC is implemented in the blacklist and scoreboard threads, along with another background thread for UNE retirement of the blacklist. The blacklist is interfaced to the Berkeley DB [36] for high level management purposes. The table size of the (single bit) hotlist is set at 232 bits, or 512 MB. The sizes of ST, AT, the graylist queue and the report queue were changed based on experimental goals. In all experiments, the sizes of AT and the graylist cache are set to be M, the age threshold.

In this experiment, only URL links are checked by PEC. An X-mark flag is inserted to the message header of a message if it has a URL on the blacklist of PEC. An UNE batch is generated using a mixture of random text files downloaded from Internet, and a spamming URL list (SUL) extracted from the spamming corpus [37]. Subject lines and sender names were randomly generated. An UNE specification for an experiment run includes the total number of messages and a list of URLs that would be selected from SUL to be put into the message body. The user can specify an exact number of times that each URL should be used and the range of the message size.

The main performance criteria include: (1) detection latency (which measures the number of messages that passes the system since onset of an UNE wave). (2) The peak throughput of the feature parser, blacklist, and scoreboard. (3) The performance impact of the queue size between the blacklist and scoreboard. (4) Finally, the impacts of the hash function on hotlist collisions, and the tradeoff between detection sensitivity, memory size and computing cycles in the scoreboard.

Figure 10 is a snapshot showing the runtime status of PEC. The four plots (starting from left upper corner, clockwise) represent UNE (red, upper half plot)/ regular (green, lower half plot) mixture of a sender source, PEC detection states, average score of all entries in ST, and AT utilization rate. The time series runs from left to right on the screen, i.e., an event at the right side of another occurred at an earlier time instance.

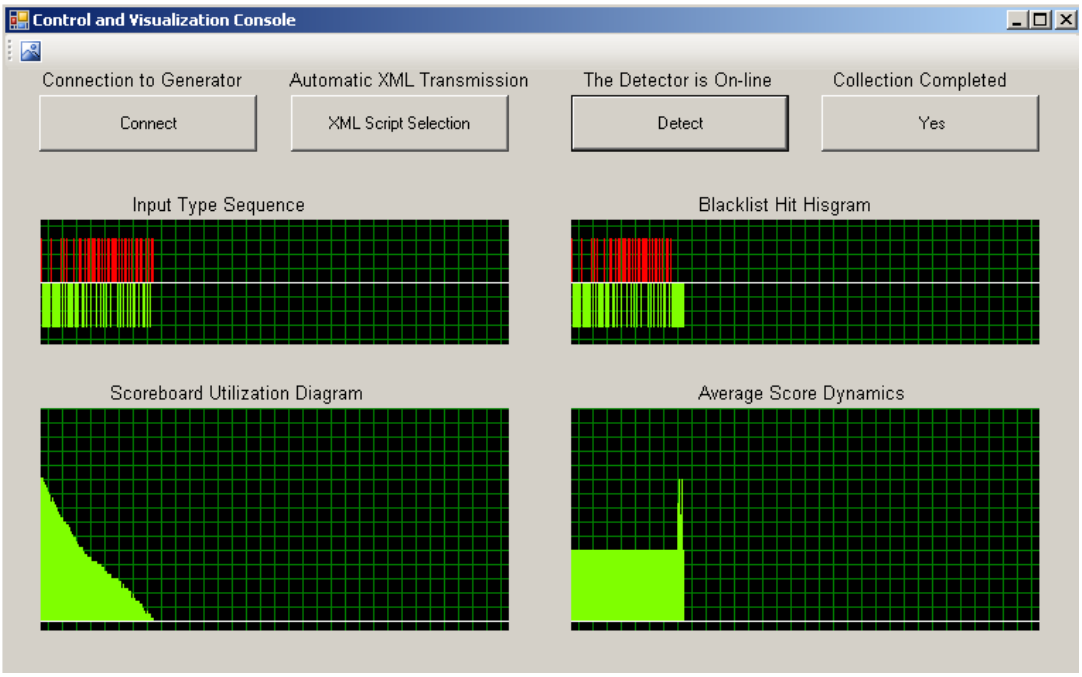


Figure 10: A snapshot of the PEC status on the control console.

Figure 11 shows the content of the AT changed when a new FI arrives at the scoreboard. In the left part of the figure, the header points to the youngest entry, [414738, 3724]. The right part shows that after a new FI [124489, 176] arrives, it becomes the youngest one, and the oldest FI [862, 1822] expires and is purged from AT and ST.

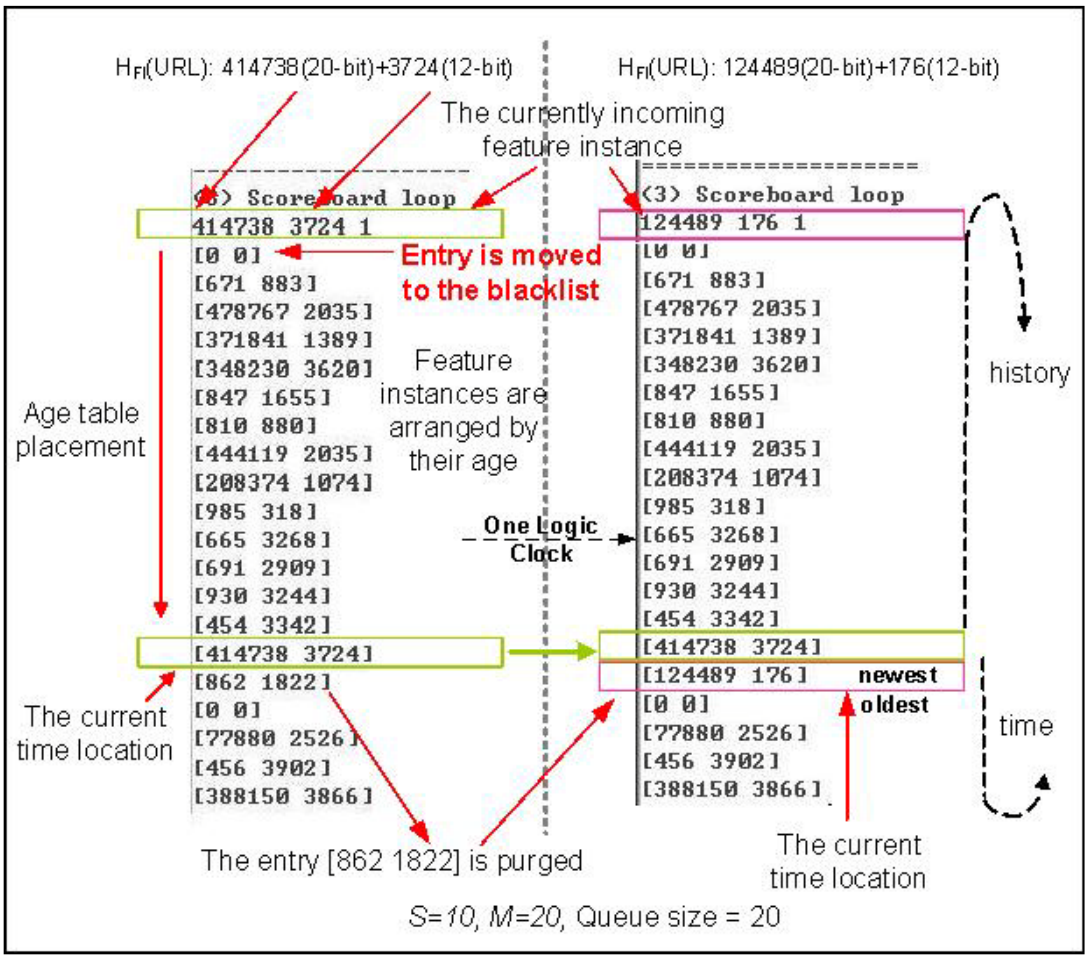


Figure 11: Scoreboard operation.

Next, we measured the detection latencies under different conditions. Let d denote the number of UNE messages divided by the number of total messages in one experiment observation bin. The expected detection latency is equal to S/d, where detection latency is the number of VCs elapsed from the first appearance of an FI until it is marked as UNE. Given a d specified by the user, UNE and regular messages are randomly placed in the bin. To eliminate potential effects of small age table size, it was always set larger than that of the observation bin, so that the relationship between

detection latency and other parameters can be characterized. In all experiments, the observation bin size was set as 2000, and every reported point was made by taking the average of 10 different runs.

3.2 Performance Evaluation

3.2.1 Detection Latency

The first experimental result is the relationship between detection latency vs. UNE density. The two curves in Figure 12 depict respectively the expected and experimental values of detection latencies of a single UNE source at six different densities (50, 100, 150, 200 …, 300 UNE/bin). The score threshold is set at 100. As expected the detection latency decreases with the UNE density. When only one UNE source is considered, a linear relationship between the scoreboard threshold and detection latency was observed.

The scoreboard is designed to detect one UNE instance at a time, and the age of every entry is affected by the densities and number of UNE sources. In the next experiment, we examined the impact of multiple UNE sources on the detection latency, where S=50. Given an UNE source A, six tests were made where one additional UNE source is added to the experiment at a time. That is, at test i, UNE A is generated with i additional UNE sources. The density of A is fixed at 100 instances per bin, where the bin size = 2000, and the density of every remaining UNE source is increased from 50 to 300 instances/bin. We observe the change of the detection latency of UNE A in the tests, and

the results are plotted in Figure 13. The last curve marked as “other sources” is the average detection latency of other non-A UNE sources.

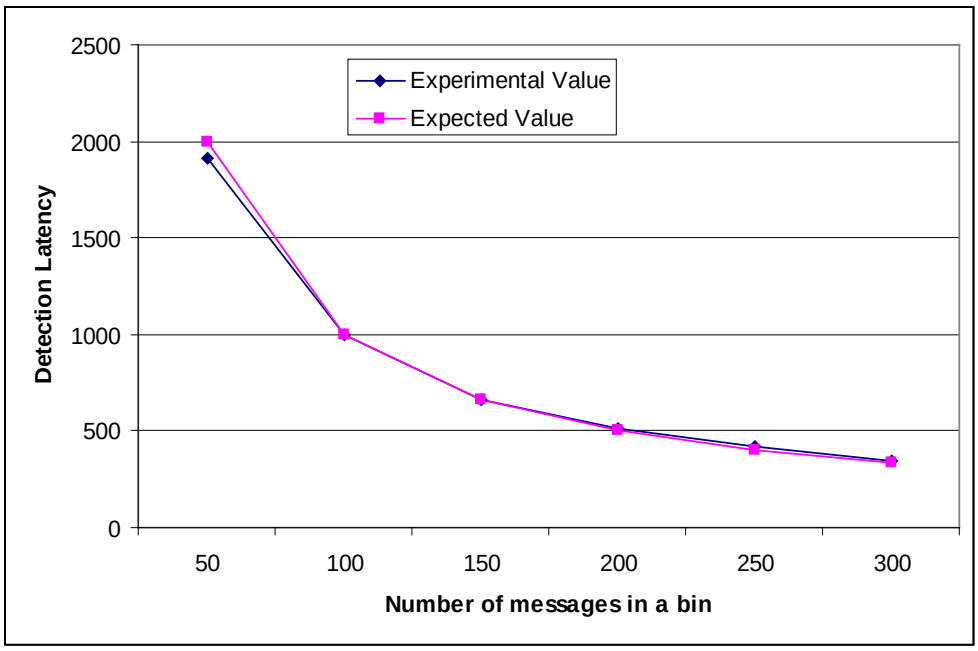


Figure 12: Detection latency of a single UNE source.

The experimental results suggest that in general, the detection latency decreases with the number of concurrent UNE sources. When A has the same density as other sources, they have the same detection latency. When A has higher or equal density as other sources, i.e., 100 vs. 50 in the first observation point, the detection latency of A is close to its expected value, i.e., 1000. The detection latency of A becomes smaller than its (single source) expected detection latency with increasing densities of other UNE sources. This is because of the vacuum effect of the VCs caused by a detected UNE. Once an UNE source is removed from the scoreboard; it will be blocked by the blacklist

indefinitely. As a result, densities of all other UNE remaining in the scoreboard are increased, and therefore detection latencies are reduced. When the density of non-A UNEs is 200 instances/bin, their measured detection latencies are 714 and 496, respectively, close to the expected values 700 and 500.

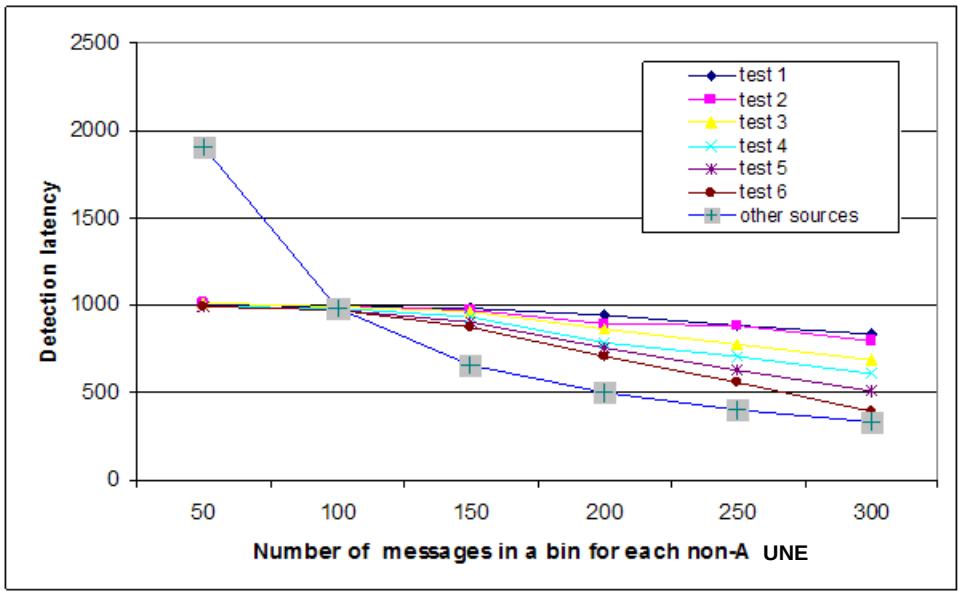


Figure 13: Detection latency for multiple UNE sources.

In addition to the detection performance, computing cost is another critical issue for PEC. For this purpose, we first evaluated different modules (feature parser, blacklist checker, and scoreboard) in isolation, and then tested the whole system with all modules integrated together.

3.2.2 Throughput of Feature Parser

The feature parser is benchmarked by message bodies embedded with randomly generated URLs. The URL parser is implemented by a light-weight deterministic finite

automaton (DFA). The input of the feature parser is the message bodies, whose average size is from 1.5 KB to 7.5 KB containing 2 URLs on average. From Figure 14 it shows that the processing capability decreases as the size of a message body increases. We did not consider MIME parsing because sendmail can separate the header and body from a message.

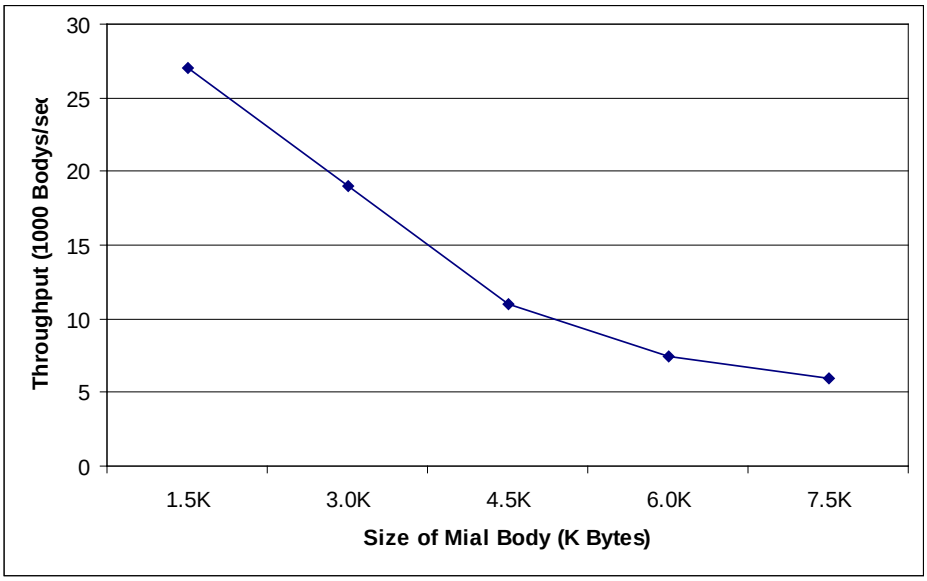


Figure 14: Throughput of feature parser.

3.2.3 Throughput of Blacklist Checking

Nowadays, URL links can contain much more than a simple domain name followed by some path. Therefore it is necessary to evaluate the sustained throughput when long and complex scripts are included in the URL. The blacklist is benchmarked by randomly generated URLs of a wide range of lengths, and the results are plotted in Figure 15. The main operations include the SDBM hash value generation, hotlist lookup, and the graylist cache update. The UNE database access is not considered in the

benchmark. The results suggest that high speed URL parsing, e.g., 300k to nearly 1M links can be parsed by commodity hardware.

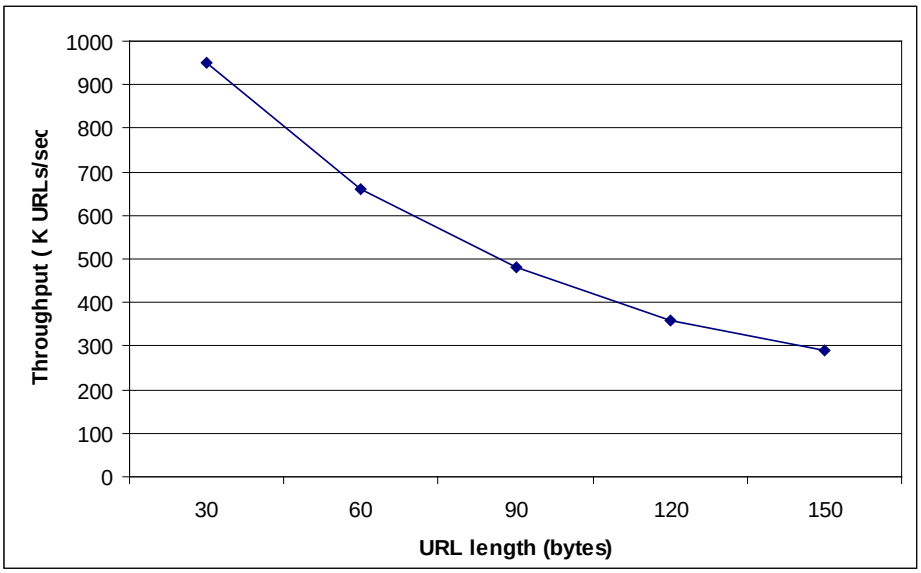


Figure 15: Throughput of the blacklist checking.

3.2.4 Scoreboard Throughput

The last, but not least throughput measure is that of the scoreboard. The scoreboard is benchmarked by randomly generated 32-bit unsigned integers, and experimental results show that it can process 1.2 M requests per second.

The two threads for blacklist and scoreboard communicate through the graylist queue and the report queue. The two threads use the mutex primitive to make mutually exclusive access to the graylist queue. As a stress test, one million hash values are directly generated by a random number generator and then fed to the blacklist and scoreboard to simulate the extreme condition that the STMP, URL parsing and hashing

were done in negligible time. Figure 16 depicts the time needed for handling 1 million hash values with change of the queue size. The results suggest that when the queue size is smaller than a threshold, the processing time would grow rapidly because of rapidly increasing overhead for the mutex operations because of inadequate queue sizes. Beyond the threshold, the queue length has minimal impact on the throughput.

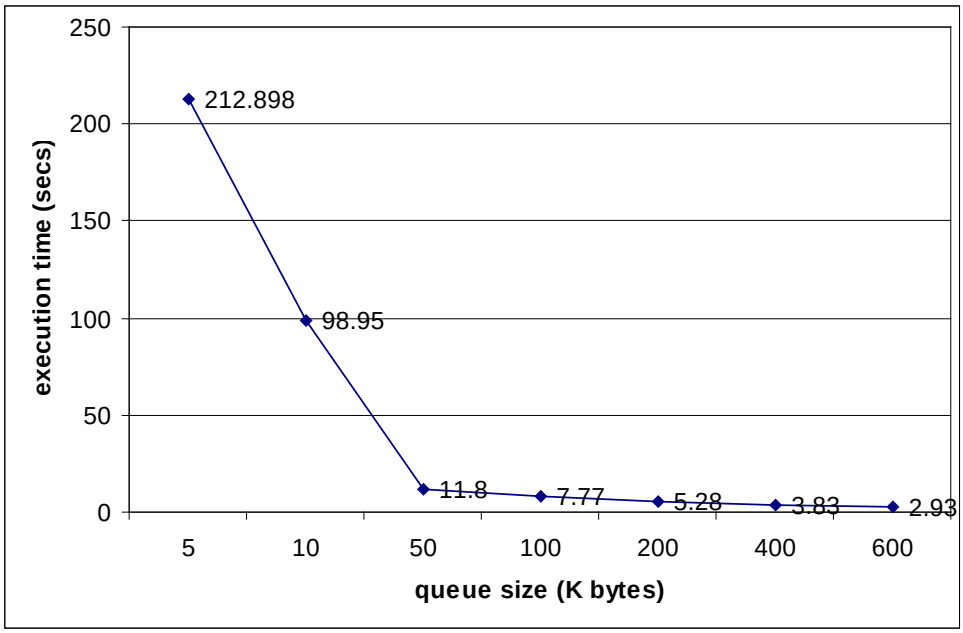


Figure 16: Scoreboard throughput vs. queue sizes

Increasing the hash length will reduce collisions but it comes at the cost of increased memory sizes. In this part of experiments, we investigated the relationship between the collisions and table sizes for the hotlist, and the relationship between detection sensitivity, sizes of ST and AT tables, and the depths of link lists in the scoreboard.

3.2.5 Collision Ratio of Black List

The hotlist is a single-bit table, so that when the hash length is 32 bits long, 512 MB of memory space suffices. Experimental results show that the collision ratio is very close to zero after 1M randomly generated URLs are hashed. The same cannot be said when we reduce the hash length, and the experimental results are given in Figure 17.

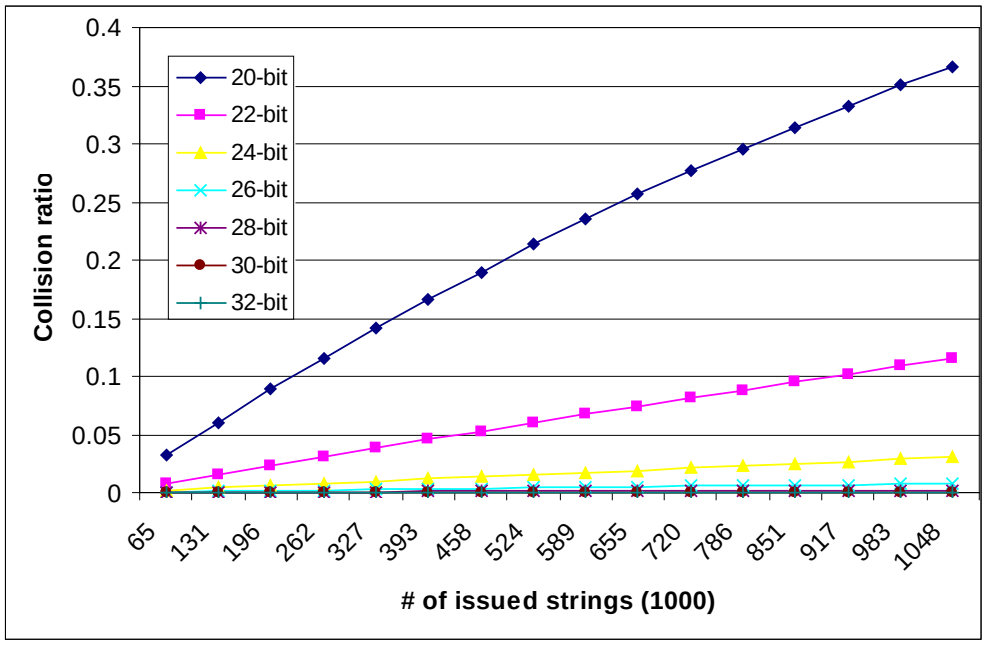


Figure 17: The collision ratio in blacklist.

There exists a more complex relationship between detection sensitivity, the sizes of AT and ST, score threshold S, and the depths of the linked lists in the ST. For simplicity, we consider S a fixed constant in this experiment. In general, we observed that increasing of M would increase UNE detection sensitivity because of longer staying of an FI in the scoreboard. As a result, the chance of hash collisions with other FIs

increases. This will lead to longer linked lists in ST. Similarly, increasing of $\boldsymbol{H_{FI}^{A}}$ (for the pointer table addressing) should reduce the depths of linked lists in ST.

To gain a sense of the relationships between these parameters, we conducted a series of experiments, where S is fixed at 50. The pointer table size is changed in different experimental runs to observe the relationship between M and the depths of linked lists. That is, given a pointer table size, the maximum depths of linked lists are measured against different M values, and the results are reported in Table 1. In this table, the most left column is the bit length of $\boldsymbol{H_{FI}^{A}}$, which also represents the size of the pointer table. For instance, when $\boldsymbol{H_{FI}^{A} = 20}$ bits, it means the pointer table has 1M entries. The top most rows represents the maximum depth of linked lists in ST, and a table entry indexed by the (table size, depth) represents the smallest M value (measured in thousands, or k) necessary to observe the said depth in ten experimental runs. For instance, if $\boldsymbol{H_{FI}^{A} = 22}$ bits, and M = 40k, one can expect the depth of linked lists in ST to be one most of the time. But if M = 50k, then the depth of some linked lists would likely be 2 or less.

Table 1: The relationship between $\boldsymbol{H_{FI}^{A}}$, M, and linked lists.

(depth)	2	3	4	5	6	7
20	20k	70k	167k	297k	582k	657k
22	45k	165k	337k			
24	145k	515k				
26	375k					
28	520k					
30	650k					

It is observed that spammers often used different URL expressions that point to the same web page, e.g., replacing the last string followed by the last "/" by another one. Tracking domain names is also a good way to differentiate UNE sources and it can significantly increase the hit ratio in the blacklist. However, when two users operate within the same domain name such as http://www.geocities.com/user1 and http://www.geocities.com/user2, using domain name alone for filtering could lead to significant false positive rates. A white list of domain names is needed to solve this problem. In this test sample, the number of tested Spammer URLs is 66501, and those are all different. Table 2 shows the result of testing how the three ways increase the hit ratio.

Table 2: Hit ratio increased by handling different prefixes.

Method	Hit Ratio Increased	Example
Remove the last character	11.9%	(1) and (2)
Remove the string after the last "/"	27.3%	(3) and (4)
Use domain name	61.2%	(5) and (6)
(1)mailshere.biz/profile/41991 (2)mailshere.biz/profile/41996 (3)appliancekiosk.geappliances.com/NASApp/ClickThru/ClickThru?q=df-1kkhQLEGhj6FEsKyiWNXKsRR (4)appliancekiosk.geappliances.com/NASApp/ClickThru/ClickThru?q=9b-7-XFQSIs_-MZdLePIxf09dRR (5)www.slammer7piggy.com/?0gQF6VyMLeif0QXXZ (6)www.slammer7piggy.com/?dXKwf0tqXJMLeiZ		

CHAPTER IV

DETECTION OF PACKET FLOW ANOMALY (ROUND TRIP TIME ESTIMATION)

4.1 Background

In networks, not all anomalies result from the content. Instead, the anomalies can exist in the rate behavior, which is not compliant to specific protocols. For flows attempting to gain more bandwidth than others, they do not necessarily follow a well-defined congestion control protocol. Flows that consume a large fraction of available bandwidth could endanger the stability of the Internet, as they frequently give a bandwidth load that is inelastic. Therefore it is a serious topic for us to minimize potential congestion collapse due to uncontrolled, high rate traffic loads. To maintain fair access of the network bandwidth, some techniques such as packet queuing and dropping can be applied to filter inelastic flows. However, no matter what kind of filter is employed, we still need to know when anomalous flows occur before they can be controlled.

The detection of flows with uncontrolled bandwidth consumption is important to cloud service providers such as Amazon EC2 and Microsoft Azure as well as content providers who are charged by the amount of the data transfer. Malicious bandwidth consumption will result in service providers and operators increasing costs to enhance network infrastructure and maintain the certain quality of service of networks.

In this chapter, I focus on detecting flows not complying with well-defined TCP protocols. TCP is a popular protocol in networks to ensure data can be delivered

correctly from client to server and avoid traffic congestion. Flows driven by unknown TCP protocols may be too aggressive in consuming the bandwidth, and thus endanger the stability of the Internet. In an enterprise gateway, two parameters that can be observed are the number of packets and inter-packet time. Unlike the TCP source, the middle point of networks is unable to know the congestion window (cwnd) size, which controls the behavior of flow rates, exactly. Since the cwnd changes every round trip time (RTT), accurate RTT estimation is an important step to infer the cwnd by counting the number of packets within a RTT. The network environment is noisy, as packets can be lost, delayed and retransmitted. It is challenging to obtain the accurate estimates of RTT and cwnd, because estimation errors can build up rapidly.

Several approaches are studied for obtaining accurate RTT estimates. I proposed the EWMA-Lomb-periodogram method to increase the accuracy of RTT estimation. The experiment outcomes show that EWMA-Lomb-periodogram is more accurate than the DFT filter bank and genuine Lomb-periodogram. EWMA-Lomb-periodogram takes unevenly spaced inter packet time as samples and applies the exponential weighted moving average to samples to reduce noises before they are fed into Lomb-periodogram. Due to the self-clocking property, the frequency resulting from RTT's will be one of the major components in the spectrum. Thus, RTT and cwnd can be estimated later.

We treat the change of the flow rate as states of the protocol behavior, which can be secured through the change of cwnd, i.e. the first order of the cwnd difference. In the regular TCP protocol, additive increase (AI) and multiplicative decrease (MD) are two major behavior states. Typically, AI is a gradual behavior, but MD occurs instantly.

Applying the low-pass filter to cwnd changes increases the risk of not detecting the occurrence of MD. On the other hand, the high-pass filter cannot reduce the noise when the flow is in the AI state. I defined 8 different behavior states and proposed using CUSUM banks to detect the flow behavior. CUSUM is not influenced by a sudden noise and its detection sensitivity can be set by the CUSUM parameters. By properly setting CUSUM parameters, we can make the CUSUM detection system not only responsive to changes but also robust to noises.

Because TCP protocols define their own state transitions, the states identified by CUSUM banks will be fed into the finite state machine (FSM), constructed based on the well defined TCP protocol to check whether a transition is legal or not. Recognizing that no estimation and detection is perfect, the FSM proposed in this dissertation is lossy and an anomaly threshold is used as a reference to count the number of aberrant transitions and that of normal transitions. Let R denotes the ratio of the number of aberrant transitions to that of normal transitions, experiment outcomes show the flows of undefined protocols can be identified based on the R-pattern. Figure 18 lists the challenges, solutions and performance criteria of detecting non-cooperative flows in networks.

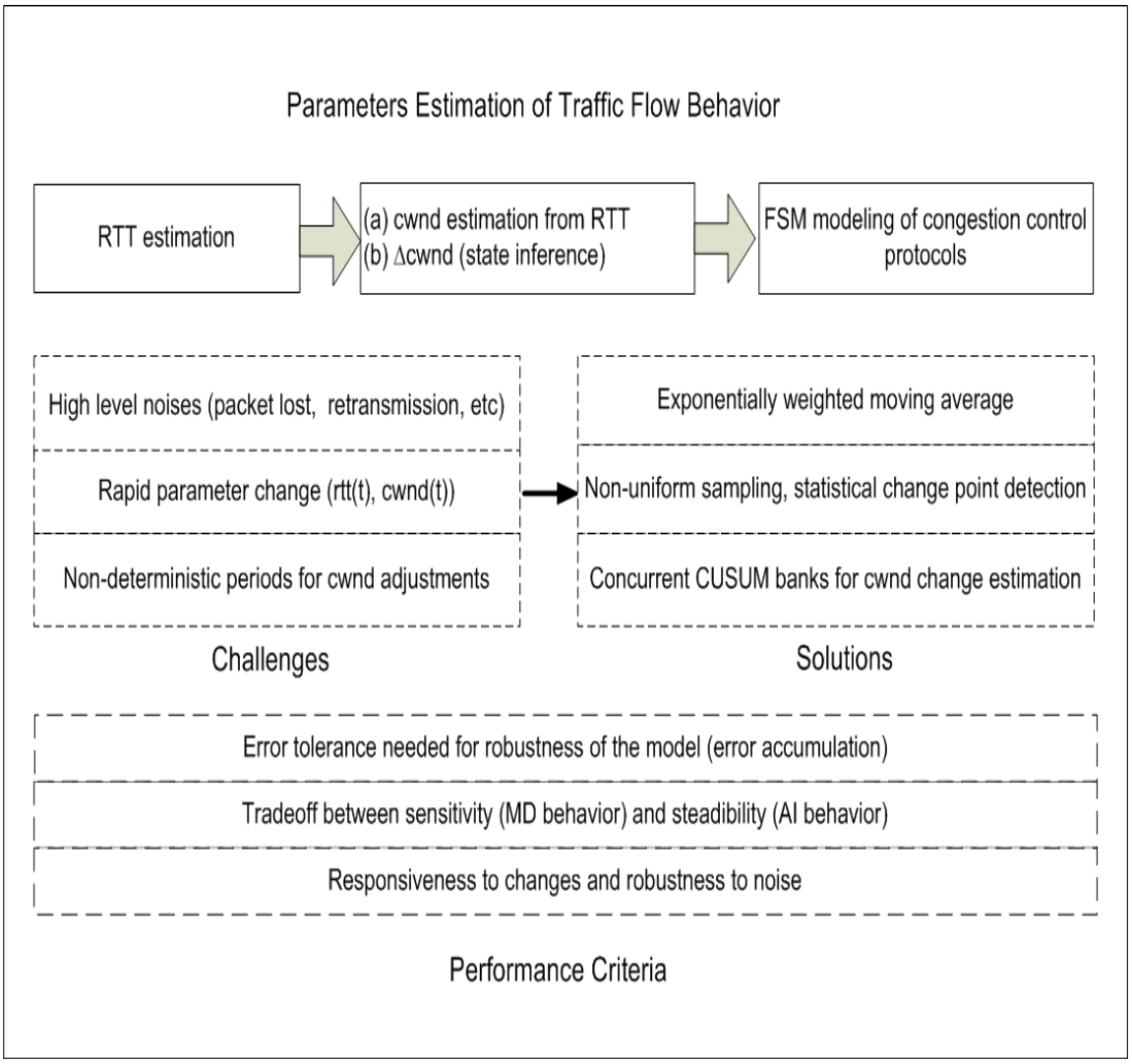


Figure 18: Challenges and solutions of anomalous flow detection.

4.2 Existing Solution

Traffic anomalies such as failures and attacks are viewed as commonplace in today's computer networks. It is a fundamental part of day-to-day network operations to identify, diagnose and treat anomalies in a timely fashion. Without this kind of capability, networks are not able to operate efficiently or reliably. [38] utilized wavelet filters, a pseudo-spline filter tuned at specific aggregation levels will expose distinct characteristics of each class of anomaly to expose the details of both ambient and anomalous traffic. [39] adopted an unsupervised machine learning technique to automate traffic classification and application identification. Because any node can send any type of traffic at any time, [40] dynamically produced hybrid traffic definitions for traffic characterization and automatically groups traffic into minimal clusters of conspicuous consumption. [41] classified Internet traffic through utilizing supervised machine learning based on a Bayesian trained neural network. [42] used the technique of observing and identifying patterns of the host behavior at the transport layer with no access to packet payload. [43] proposed adaptive NetFlow, to address many shortcomings of NetFlow by dynamically adapting the sampling rate to achieve robustness without sacrificing accuracy. [44] presented the first large-scale exploration of the power of the subspace method which characterizes network-wide anomalies by fusing information from flow measurements taken throughout a network. For anomaly detection for large scale networks, a Kalman filter can be used to filter out the “normal” traffic [45]. Principal component analysis was used to diagnose traffic anomalies [46], to accurately detect when a volume anomaly is occurring, and correctly identified the

underlying origin-destination flow which is the source of the anomaly. Two unsupervised clustering algorithms, namely K-Means and DBSCAN [47] were proposed for network traffic classification. The experimental results indicated that DBSCAN produces better clusters; although, DBSCAN has lower accuracy compared to K-Means. [48] fed hand-classified network data to a supervised Naive Bayes estimator and showed it is able to achieve about 65% accuracy on per flow classification.

Several methods were proposed to deal with bandwidth consumption issues. Stabilized RED, SRED [49] was proposed to maintain the flow fairness and also identified flows that may be misbehaving, i.e. taking more than their fair share of bandwidth. A game theoretic framework [50] was used to detect distributed bandwidth attacks with needed sampling rates. Different from above approaches, our goal is to classify flows of different protocols based on their elasticity, which is the critical concept of this dissertation, with the minimum false rate.

[51] showed that using feature distributions, anomalies naturally fall into distinct and meaningful clusters. These clusters can be used to automatically classify anomalies and to uncover new anomaly types. A simple architecture called P4P was proposed in [52] to allow more effective cooperative traffic control between applications and network providers and thus achieve efficient and fair utilization of Internet network resources. [53] proposed a way to detect abrupt changes in the network traffic based on change-point detection theory and utilized a threshold of test statistics to achieve a fixed rate of false alarms.

4.3 Limitation of Statistical Models to Detect Non-cooperative Flows

Non-cooperative TCP flows can be modeled from different aspects. One option is to model abnormal TCP flows as being selfish, where they do not reduce their flow rate when the network is congested.. To translate this seemingly simple observation into quantified, measurement based assessments for on-line traffic, we first give some basic insights on the relationship between different parameters in a well know traffic model [54]. Here, given a time interval $\boldsymbol{\tau}$ seconds and normal TCP flow throughput $\bar{\boldsymbol{\lambda}}$ pkts/sec, a TCP flow is regarded anomalous if its throughput $\lambda > \hat{\boldsymbol{\lambda}} + \boldsymbol{\kappa}$, where $\boldsymbol{\kappa}$ is the excessive throughput and $\hat{\boldsymbol{\lambda}}$ as shown in (13) is the long-term steady-state TCP throughput of the regular TCP flow.

$$\hat{\boldsymbol{\lambda}} = \mathbf{min}\left(\frac{\mathbf{W_{max}}}{\mathbf{RTT}}, \frac{\mathbf{1}}{\mathbf{RTT}\sqrt{\frac{\mathbf{2bp}}{\mathbf{3}}} + \mathbf{T_0}\,\mathbf{min}\left(\mathbf{1}, \mathbf{3}\sqrt{\frac{\mathbf{3bp}}{\mathbf{8}}}\right)\mathbf{p}(\mathbf{1} + \mathbf{32p^2})}\right) \qquad (13)$$

It is straightforward to observe that when TCP flows unfairly consume the network bandwidth, $\boldsymbol{\kappa}$ is proportional to $\boldsymbol{\tau}$. However, equation (13) is the mathematical model of the TCP throughput in the steady state, but most of its variable values, i.e., maximum window size $\mathbf{W_{max}}$, the round trip time **RTT**, the bottleneck bandwidth $\mathbf{b}$ and the packet loss probability $\mathbf{p}$, cannot be measured at an enterprise gateway for a large number of flows. In addition, the steady state model is incapable of detecting the change of transient behaviors timely. Due to aforementioned reasons, the statistical model has its limitation for identifying anomalous packet flows.

A good indicator to show the TCP transient behavior is the congestion window, cwnd, the parameter that TCP utilizes to avoid the network congestion. Unfortunately, this parameter cannot obtained from throughput based measurements of TCP flows at a measure point, because it is not any part of Equation (13). To attack this problem, we will divide the traffic flow analysis problem into multiple stages, each of which is responsible for acquiring certain parameters, in order to infer TCP flows' behaviors. To facilitate the discussion, we first summarize major terminologies and their symbols in Table 3 below.

Table 3: Equation symbols.

Symbol	Definition
$\mathbf{F_i}$	the i-th flow
$\mathbf{W_i(t)}$	congestion window of $\mathbf{F_i}$ at time t
$\mathbf{q(t)}$	bottleneck queue length at time t
L	upper bound of $\mathbf{l(t)}$
$\mathbf{RTT_i}$(t)	round trip time of the flow $\mathbf{F_i}$ at time t
$\mathbf{R_i(t)}$	rate of the flow $\mathbf{F_i}$ at time t
$\mathbf{R(t)}$	total incoming packet rate at time t
$\mathbf{N_L}$(t)	total number of packets lost at a location.
$\mathbf{P_i(t)}$	probability of the packet lose of $\boldsymbol{F_i}$ at time t

Next, we observe basic challenges in modeling of TCP flow behaviors based on packet losses. Mathematically, the number of output packets is equal to that of input

packets plus the number of lost packets and remaining usable queue capacity. That is, one can have

$$\boldsymbol{N_L(t) = \max\left(0, \int_{\tau=t}^{t+\Delta t}(R(\tau) - B)d\tau\right)}\text{-(L-q(t)))} \quad (14)$$

By substituting $\mathbf{R(\tau) = \sum_i (1 - P_i(\tau)) W_i(\tau)/RTT_i(\tau)}$ into Equation (14), we obtain

$$\boldsymbol{N_L(t) = \max\left(0, \sum_i \int_{\tau=t}^{t+\Delta t}(1 - P_i(\tau)) W_i(\tau)/RTT_i(\tau) - B)d\tau\right)}\text{-(L-q(t))).} \quad (15)$$

To obtain $\mathbf{\sum_i W_i(t)}$, we need to know the parameters: $\mathbf{RTT_i}$ as well as the number of packets lost at the bottleneck and during the transmission. From the aforementioned discussion, we conclude that how to make precise, robust estimation on RTT for TCP flows at a gateway is the foundation to classification of congestion control behaviors. In this discussion, we mainly focus on congestion control behaviors depicted in **Figure 19**; They are AIAD (additive increase, additive decrease), MIMD (multiplicative increase, multiplicative decrease), AIMD (additive increase, multiplicative decrease), and MIAD(multiplicative increase, additive decrease).

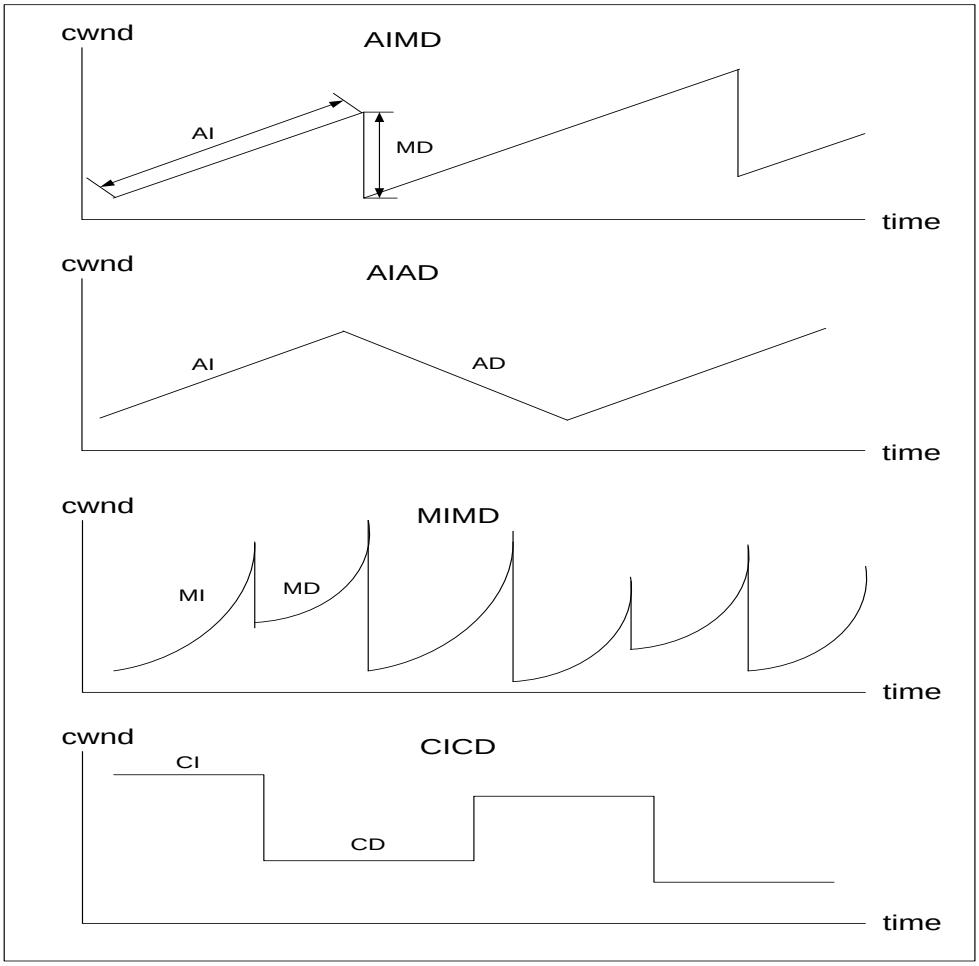


Figure 19: Congestion window behaviors.

4.4 Round Trip Time (RTT) Estimation

To infer $\mathbf{cwnd_i(t)}$ of $\mathbf{F_i}$, one is able to count the number of packets every fixed interval. Let $\mathbf{CC}$ denote the cumulative counter. Then the AI behavior can be identified by checking if $\mathbf{CC[n+1] - CC[n] = N^2}$ assuming the sampling time is exactly equal to the N RTTs, where n indexes the sample number. However, this assumption is not true because RTTs are variable and the boundary of two RTTs should be identified clearly. Hence, if the sampling interval is less than one RTT, then the difference between

$\mathbf{CC[n+1]\ and\ CC[n]}$ can be negative. In addition, $\mathbf{CC[n+1]-CC[n]=N^2}$ cannot hold if the sampling period cannot correctly respond to the physical $\mathbf{RTT_i}$(t). Furthermore, the RTTs of all flows are not identical. It is infeasible to identify all packets within the same window just by counting packets in a fixed time interval.

Figure 20 is an example showing that the incorrect boundary of RTTs can lead to incorrect cwnd estimates. In an AI state, the cwnd value will increase by 1 every other RTT, i.e. $\mathbf{cwnd(RTT_{i+1})-cwnd(RTT_i)=1}$. However, $\mathbf{CC[n+1]-CC[n]\neq 1}$ if the burst size is overestimated or underestimated in the AI state. Hence, the ituitive packet counting without the RTT knowledge will lead to the wrong inference of the TCP behavior.

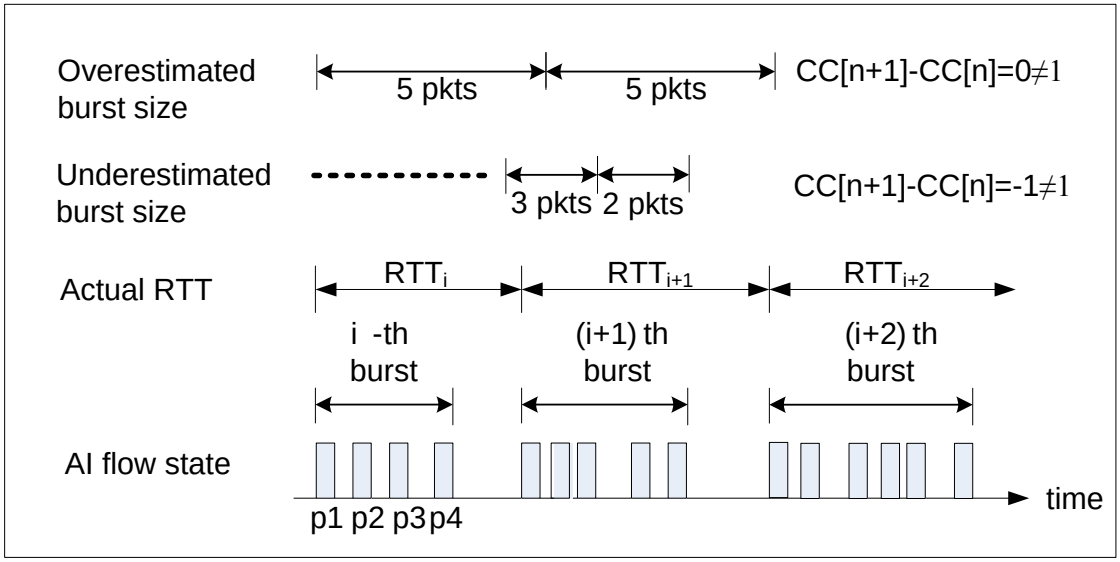


Figure 20: Incorrect window estimates based on incorrect RTT boundary.

In order to obtain the cwnd value, acceptable RTT estimation plays the critical role. In the following sections, both DFT and Lomb periodgram are studied to verify the accuracy of RTT estimates.

4.5 DFT-based RTT Estimation

Rather than the CC method, the discrete Fourier transform, DFT, is one of the options to capture the frequency of the self-clocking property of TCP flows. The above diagram shows a system is designed to obtain the major frequency components and derive the round trip time dynamically. L denotes the window size, which is set to 256 by default. The i-th sampling frequency $\boldsymbol{F_s}$ is derived by Equation (16), where $\mathbf{IPT_k}$ means the k-th inter packet time.

$$\boldsymbol{F_s(i) = L / \sum_{k=i}^{i+L-1} IPT_k} \tag{16}$$

Here the sampling intervals in DFT are assumed even though no such of a sampler is used to capture IPTs. Based on Equation (16), the average inter packet time X(n) is expressed as

$$\boldsymbol{X(n) = 1 / F_s(n)} \tag{17}$$

Figure 21 shows that L samples are fed into the uniform DFT filter banks, of which the decimation coefficient is M. If M=2m, then DFT can be performed by fast Fourier Transform (FFT). Given $\mathbf{H(z)}$ is the transfer function of the casual low-pass digital filter, then $\mathbf{H_0(z) = H(z)}$ denotes the prototype filter and $\mathbf{H_k(z)}$ the k-th band filter. Then we have $\mathbf{H_k(z) = H_0(z)(zW_k)}$, where $\mathbf{W = e^{-j2\pi/M}}$.

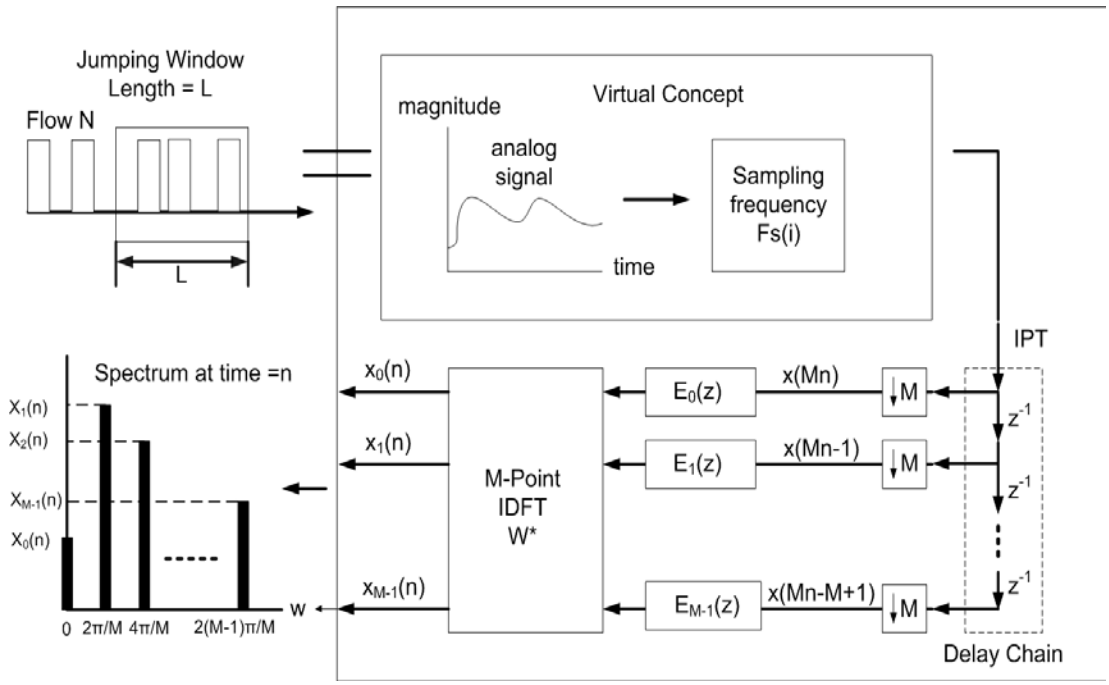


Figure 21: RTT estimator – uniform sampling

Let N denote the order of the low pass filter. Given

$$\boldsymbol{H(z) = \sum_{k=0}^{M-1} z^{-k} E_k(z^M)} \tag{18}$$

where $\boldsymbol{E_k}$(z) is the k-th polyphase component of H(z). The decimated DFT filter bank needs $\mathbf{\frac{M}{2} logM + N}$ multiplers which is much less than NM multipliers for computation. The higher order of $\mathbf{E_k(z)}$ will lead to the sharper cutoff frequency and higher stopband attenuation

$$\begin{bmatrix} \boldsymbol{H_0(Z)} \\ \boldsymbol{H_1(Z)} \\ \boldsymbol{H_2(Z)} \\ \vdots \\ \boldsymbol{H_{M-1}(Z)} \end{bmatrix} =$$

$$\begin{bmatrix} \mathbf{1} & \mathbf{1} & \mathbf{1} & \cdots & \mathbf{1} \\ \mathbf{1} & \boldsymbol{W^{-1}} & \boldsymbol{W^{-2}} & \cdots & \boldsymbol{W^{-(M-1)}} \\ \mathbf{1} & \boldsymbol{W^{-2}} & \boldsymbol{W^{-4}} & \cdots & \boldsymbol{W^{-2(M-1)}} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{1} & \boldsymbol{W^{-(M-1)}} & \boldsymbol{W^{-2(M-1)}} & \cdots & \boldsymbol{W^{-(M-1)^2}} \end{bmatrix} \begin{bmatrix} \boldsymbol{E_0(Z^M)} \\ \boldsymbol{z^{-1}E_1(Z^M)} \\ \boldsymbol{z^{-2}E_2(Z^M)} \\ \vdots \\ \boldsymbol{z^{-(M-1)}E_{M-1}(Z^M)} \end{bmatrix}, \quad (19)$$

Equation (19) shows the transfer function of the DFT filter bank is equal to the inverse DFT matrix multiplying delayed input signals. The output of the transfer function is the spectrum that represents the amplitudes of all frequency components.

To verify the RTT estimation outcome from DFT, an experiment based on NS2 is configured as shown in Figure 22, where 50 TCP-Reno flows are set to run in 60 seconds. The bandwidth and delay for the link $\mathbf{S_i - O_1}$ and $\mathbf{O_2 - R_j}$ are 10 Mb/s and 10 ms respectively. For the link $\mathbf{O_1 - O_2}$, its bandwidth is 100Mb/s and delay is 20 ms. The sampling frequency is derived based on Eq. (16). The DFT obtains a sequence of RTTs by computing the spectrum every 256 input samples.

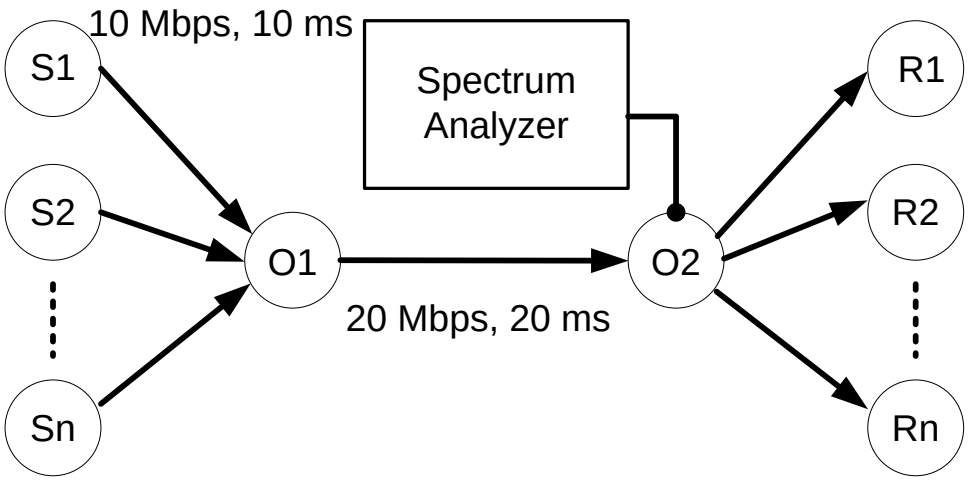


Figure 22: Network topology of RTT estimation.

Figure 23 shows the spectrum of two TCP flows using DFT. In this dissertation, the k major frequency components are defined as those with the largest k amplitudes. Among major frequencies, the lowest one is selected to compute the estimated RTT because the frequency of the TCP self-clocking is lower than that of network noises. In the experiment, k is set 3.

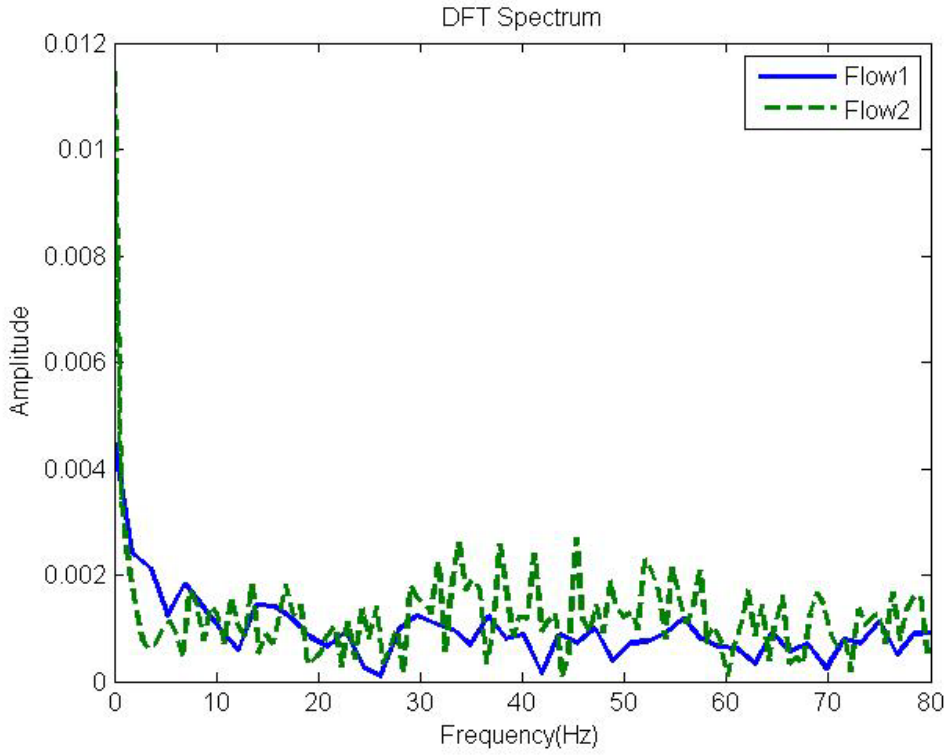


Figure 23: Spectrum of two TCP flows.

Figure 24 plots the RTT estimates, which are the inverse of the lowest major frequency and the real RTTs, which are obtained from the NS2 trace variable rtt_. The real RTTs expressed by the green star dynamically change in the range [80 90] ms, However, most of RTT estimates represented by the red square are much larger than real ones.

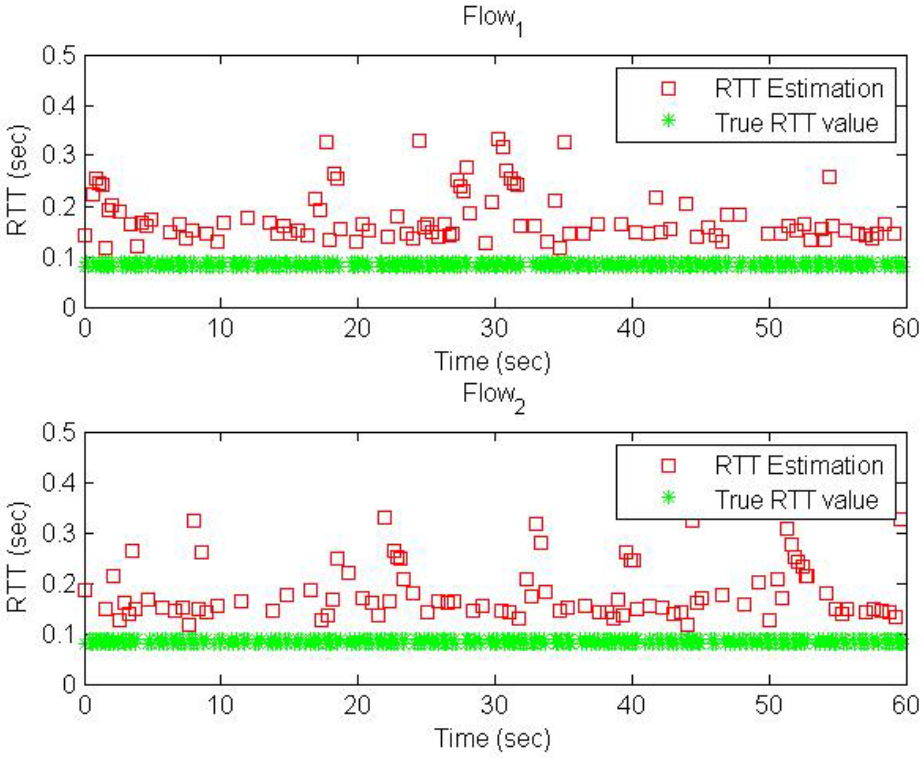


Figure 24: RTT estimates and real RTTs.

4.6 EWMA Lomb-based RTT Estimation

Since the inter packet time is not evenly spaced and the boundary of packets of two adjacent congestion windows is influenced by network uncertainties such as packet loss, queuing dynamics and packet retransmission, it is almost infeasible to identify each $\mathbf{R_i(t)}$ with 100% accuracy to estimate the congestion window size $\mathbf{cwnd_i(t)}$ as the reference of the flow behavior.

Instead of counting the number of burst packets using the fixed sampling period, we recognize the time-varying property of RTT, which is regarded as a constant in most TCP models. We employ the Lomb periodogram [54][55][56] to obtain the power

spectral density (PSD) estimates directly from irregularly sampled time series such as the inter packet time, without resampling, to dynamically estimate uncertain RTT values, which are key factors to distinguish two adjacent bursts in the measure point. Packets are grouped to a burst if they are sent within the same congestion window.

Lomb periodogram is an approach used to analyze data with highly irregular sampling, especially if there are long stretches without data. Lomb periodogram is better than the power spectrum for bottleneck links where there would be a retransmission timeout such that the sender has no data to transmit before the timer expires. Given the time series $\{\mathbf{x_i}\}$ measured at times $\mathbf{t_i}, \mathbf{i = 1, 2 \ldots n}$, with mean $\bar{\mathbf{x}}$ and standard deviation $\boldsymbol{\sigma_x}$,

$$\mathbf{L(\omega)} = \mathbf{1/2}\sigma_x^2 \left\{ \frac{\left(\sum_i (x_i - \bar{x}) cos\, \omega\left(t_i - \tau(\omega)\right)\right)^2}{\sum_i cos^2\, \omega\left(t_i - \tau(\omega)\right)} + \frac{\left(\sum_i (x_i - \bar{x})\, sin\, \omega\left(t_i - \tau(\omega)\right)\right)^2}{\sum_i sin^2\, \omega\left(t_i - \tau(\omega)\right)} \right\} \tag{20}$$

and

$$\boldsymbol{\tau(\omega)} = \frac{\mathbf{1}}{\mathbf{2\omega}} \mathbf{tan^{-1}} \frac{\sum_i \mathbf{sin\, 2\omega t_i}}{\sum_i \mathbf{cos\, 2\omega t_i}}. \tag{21}$$

Here $\boldsymbol{\tau(\omega)}$ is an offset that makes $\mathbf{L(\omega)}$ invariant to time translation. Therefore, we can consider the Lomb periodogram as a method to fit a sinusoid of angular frequency $\boldsymbol{\omega}$ to the data [57] based on linear least squares. Hence, for each $\boldsymbol{\omega}$, Equation (20) is equivalent to fitting data into the mode

$$\mathbf{x(t)} = \mathbf{a\cos(\omega t)} + \mathbf{b\sin(\omega t)}, \quad (22)$$

based on the least-square criteria and $\mathbf{L(w)} = \mathbf{a^2} + \mathbf{b^2}$. The algorithm to compute the Lomb periodogram makes indirect use of the FFT and requires O(n log n) operations. Let $\mathbf{w}^*$ be the angular frequency that maximizes $\mathbf{L(w)}$ on a suitable grid over the interval, then the estimate $\widehat{\boldsymbol{\theta}}$ of the RTT is defined by

$$\widehat{\boldsymbol{\theta}} = \frac{\mathbf{2\pi}}{\mathbf{w}^*}, \quad (23)$$

It is common that the measured data points are the sum of a periodic signal and independent Gaussian noise. If we are trying to determine the presence of such a periodic signal, we want to be able to understand "How significant is a peak in the spectrum$\mathbf{L(\omega)}$?" The null hypothesis in the question is that the data values are independent and follow the Gaussian distribution. In other words, null hypothesis here means the significance level of any peak in $\mathbf{L(w)}$ that we can see.

Another nice property of the Lomb normalized periodogram is that the viability of the null hypothesis can be tested fairly rigorously [55]. In the case of the null hypothesis, $\mathbf{L(\omega)}$ has an exponential probability distribution with unit mean. Given $\boldsymbol{\omega}$ and $\mathbf{L(\omega)} = \mathbf{z}$ in M independent frequencies, the probability of none values greater than z is $(\mathbf{1} - \mathbf{e^{-z}})^{\mathbf{M}}$ such that

$$\mathbf{P}(> \boldsymbol{z}) = \mathbf{1} - (\mathbf{1} - \mathbf{e^{-z}})^{\boldsymbol{M}}, \quad (24)$$

is the false-alarm probability of the null hypothesis [55]. A small value of P for the false-alarm probability indicates a highly significant periodic signal. In general M depends on the number of frequencies sampled, the number of data points N, and their detailed

spacing. If the data points are approximately equally spaced, M is very nearly equal to N.

Figure 25 is the Lomb-based RTT estimation system consisting of several components as follows: (1) Sample feeding, (2) EWMA IPT smoothing, (3) Enough number of samples, (4) Flow ID dispatching and threshold signaling, (5) Block data copy into ring buffer, (6) Run Lomb periodgram algorithm, (7) Retrieve the major frequency components and (8) Obtain RTT estimates of each flow.

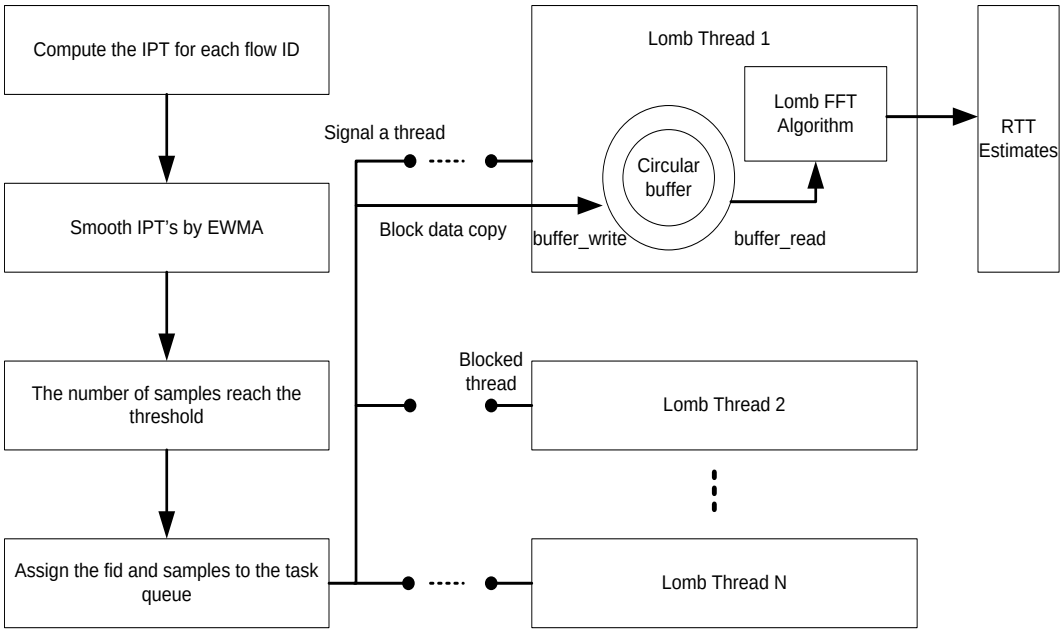


Figure 25: A prototype architecture of the Lomb-based RTT estimation system.

Two key parameters of the Lomb periodogram are *oversampling frequency*, ofac, and *high frequency*, hifac, are set as 4 and 2, respectively. The number of different frequencies $\mathbf{N_p}$ returned by the program is then given by

$$N_p = \frac{ofac * hifac}{2} * N, \tag{25}$$

where N is the number of data samples.

The experimental topology, which consists of 50 source nodes, 2 router nodes, and 50 receiver nodes, is the same as in Figure 22. The transmission delay is 10 ms between $\mathbf{S_i}$ and $\mathbf{O_1}$, 20 ms between $\mathbf{O_1}$ and $\mathbf{O_2}$, as well as 10 ms between $\mathbf{O_2}$ and $\mathbf{R_i}$, where $\mathbf{i = 1, 2, \dots, 10}$. The bandwidth of all links is set to 5 Mbps except that the link $\mathbf{O_1 - O_2}$ is 100 Mbps. Each $\mathbf{S_i}$ delivers a TCP-Reno flow to the destination $\mathbf{R_i}$ over the patch of $\mathbf{S_i - O_1 - O_2 - R_i}$. IPT's of all flows are collected at $\mathbf{O_2}$.

There exist a large number of noises in the IPT sequence such that the amplitudes of noise frequencies are non-neglected in the Lomb-periodogram. To eliminate noises in the IPT sequence, an exponentially weighted moving average, EWMA, $\mathbf{IPT_{ewma}(t + 1) = (1 - \alpha) * IPT_{ewma}(t) + \alpha * IPT(t)}$ is proposed in this work to filter signals of each flow. The EWMA approach is widely adopted in the network congestion control, e.g. RFC2988 applies EWMA to compute the smooth RTT. The EWMA-Lomb approach leads to more accurate and stable RTT estimation compared to the regular Lomb approach.

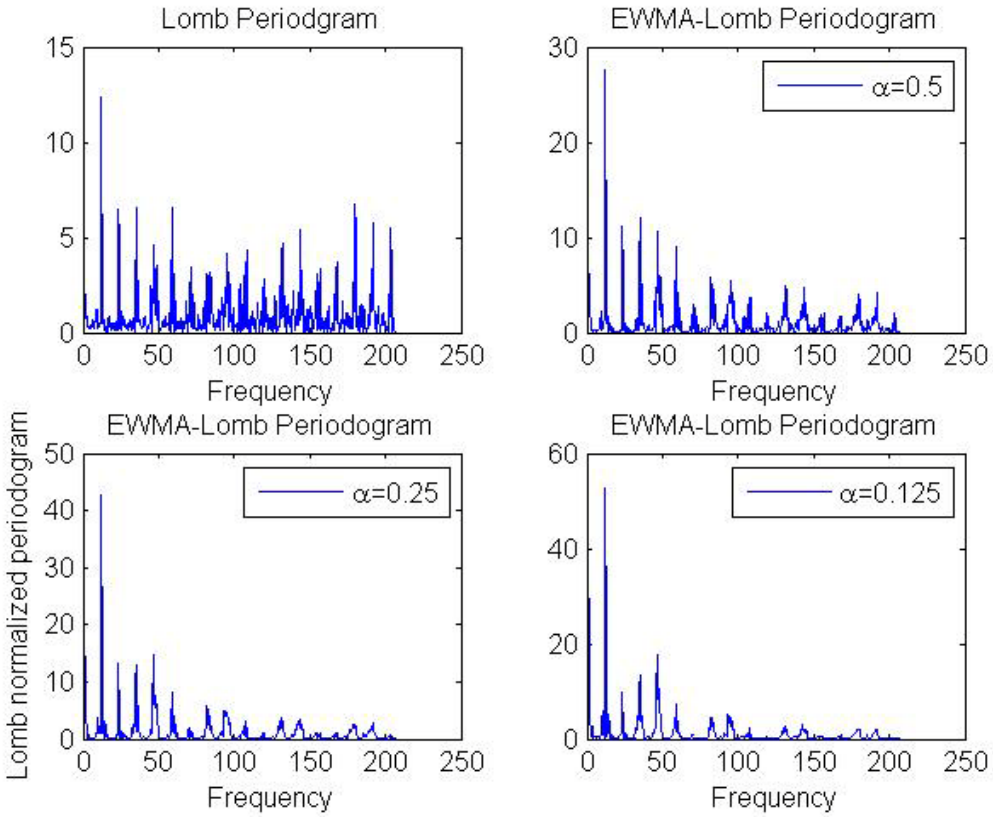


Figure 26: Lomb periodgrams.

Figure 26 shows the snapshot of Lomb periodogram of one of 50 simulated TCP flows based on the regular Lomb periodogram and EWMA Lomb periodogram with different weighting factors from $\boldsymbol{\alpha = 0.5, \alpha = 0.25}$ to $\boldsymbol{\alpha = 0.125}$. In the diagram, the regular Lomb plot cannot generate strong amplitude of the self-clocking frequency, 12.37 Hz, compared to those of noises. However, EWMA Lomb periodogram tends to enhance the major frequency components and suppress noisy components when $\boldsymbol{\alpha}$ becomes smaller. Once the self-clocking frequency is obtained, the estimated RTT is therefore equal to the inverse of the self-clocking frequency. Given a Lomb

periodogram, the major frequency components are defined as those with largest k amplitudes and their false alarm probability is less than a predefined value p. In this dissertation, k is set to 3 and p 0.01%. Among major components, the one with the lowest frequency is considered related to RTT due to the TCP self-clocking property, while other high frequency ones are due to network noises such as packet loss, packet retransmission and queuing delay.

Figure 27 shows RTT estimates of a NS2 trace using Lomb and EWMA Lomb approaches respectively. The green marks are real time RTT's recorded in NS2 and red marks are RTT estimates based on the estimation. The real RTT's are mostly between 80 ms or 90 ms, except for the simulation time between 0.69 seconds and 2.15 seconds for which RTT's are in between 0.11 and 0.14 seconds.

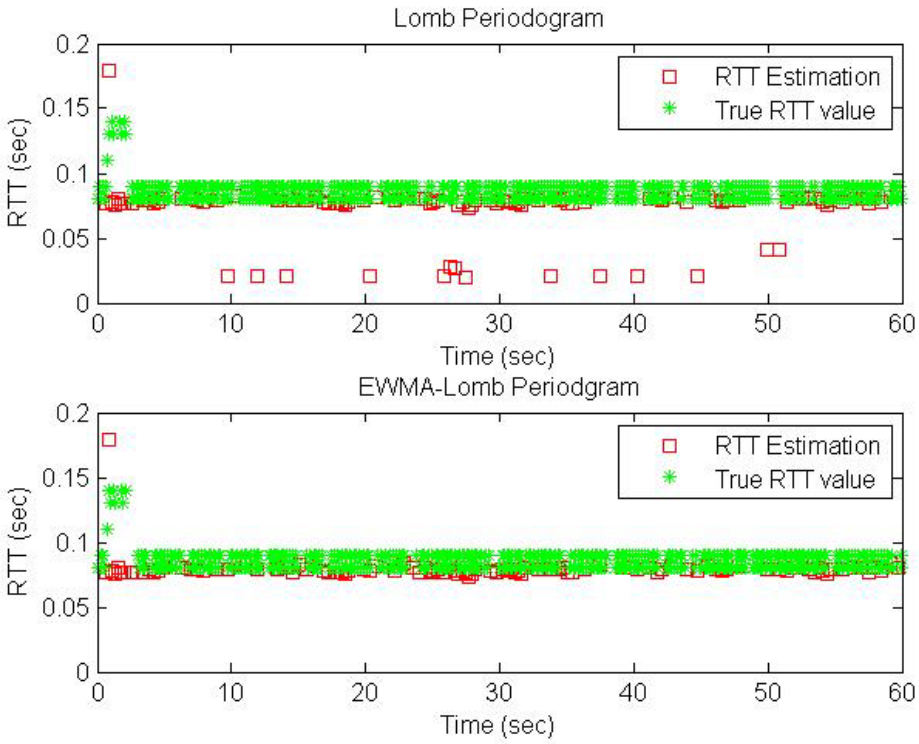


Figure 27: Lomb-based RTT estimates for a trace.

Figure 27 also shows that RTT estimates based on the EWMA-Lomb are resistant to the noises and close to the real RTT values. However, the regular Lomb method may generate RTT estimates far away from the real one due to the influence of high frequency noises. Based on the diagram, most of EWMA Lomb based RTT estimates are very close to the real RTTs although a few outliers can still be observed. Because no estimation is 100% perfect, we wonder the error rate of RTT estimates based on Lomb and EWMA Lomb respectively.

To understand the estimation error, let root-mean-square error, RMSE, be defined as follows

$$\mathbf{RMSE}(\widehat{\boldsymbol{\theta}}) = \sqrt{\mathbf{E}[(\widehat{\boldsymbol{\theta}} - \boldsymbol{\theta}(\boldsymbol{t}))^{\mathbf{2}}]}, \tag{26}$$

where $\widehat{\boldsymbol{\theta}}$ is the estimator with respect to the estimated parameter $\boldsymbol{\theta}$. Since the real RTT changes dynamically and the number of real RTTs from the trace files is more than that of RTT estimates, $\overline{\boldsymbol{\theta}}$, the mean of real parameters between two parameter estimates, is used instead of $\boldsymbol{\theta}$ in Equation (26). Let $\widehat{\mathbf{RTT}}$ denote the RTT estimator, $\mathbf{RTT_i}$ denote the i-th RTT estimate at time $\mathbf{t_i}$, and $\mathbf{RTT(t_j)}$ the real RTT at time $\mathbf{t_j}$ then

$$\mathbf{RMSE}\left(\widehat{\mathbf{RTT}}\right) = \sqrt{\mathbf{E}\left[\left(\mathbf{RTT_i} - \int_{\mathbf{t=t_{i-1}}}^{\mathbf{ti}} \frac{\mathbf{RTT(t)}}{\mathbf{(t_i - t_{i-1})}}\mathbf{dt}\right)^{\mathbf{2}}\right]}, \tag{27}$$

In Figure 28, RMSE's of 10 flows based on EWMA Lomb are smaller than those of pure Lomb for the same trace.

Let $\mathbf{P_{err}}$ denote the error percentage such that

$$\boldsymbol{P_{err}} = \mathbf{E}\left[\frac{\mathbf{abs}(\widehat{\boldsymbol{\theta}} - \overline{\boldsymbol{\theta}}(\boldsymbol{t}))}{\overline{\boldsymbol{\theta}}(\boldsymbol{t})}\right] * \mathbf{100}, \tag{28}$$

Figure 29 shows the $\mathbf{P_{err}}$ of Lomb-based RTT estimates and that of EWMA-Lomb RTT estimates for 10 out of 50 TCP flows. The mean of $\mathbf{P_{err}}$ of TCP Reno flows is 24% and 10% for regular Lomb and EWMA Lomb respectively. Hence the RTT estimation based on the EWMA IPTs can result in acceptable RTT estimation errors.

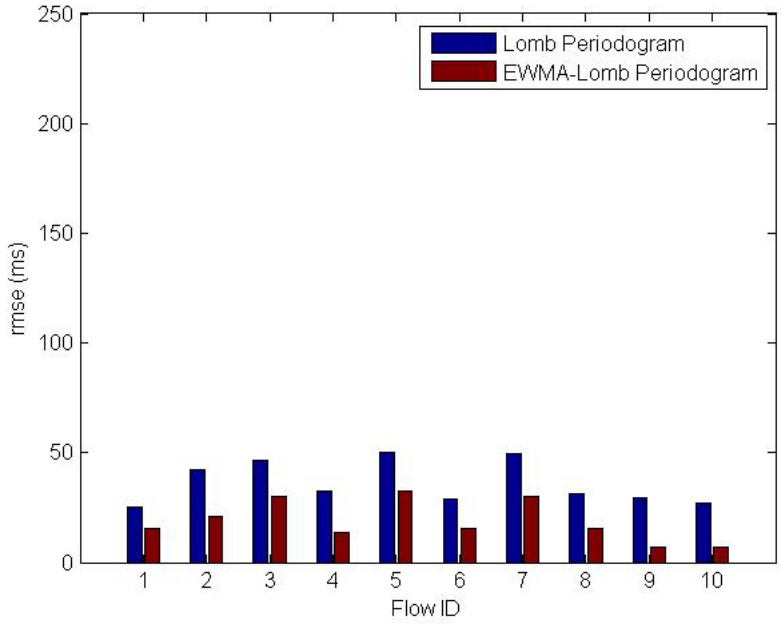


Figure 28: RMSE of RTT for 10 of 50 simulated TCP flows.

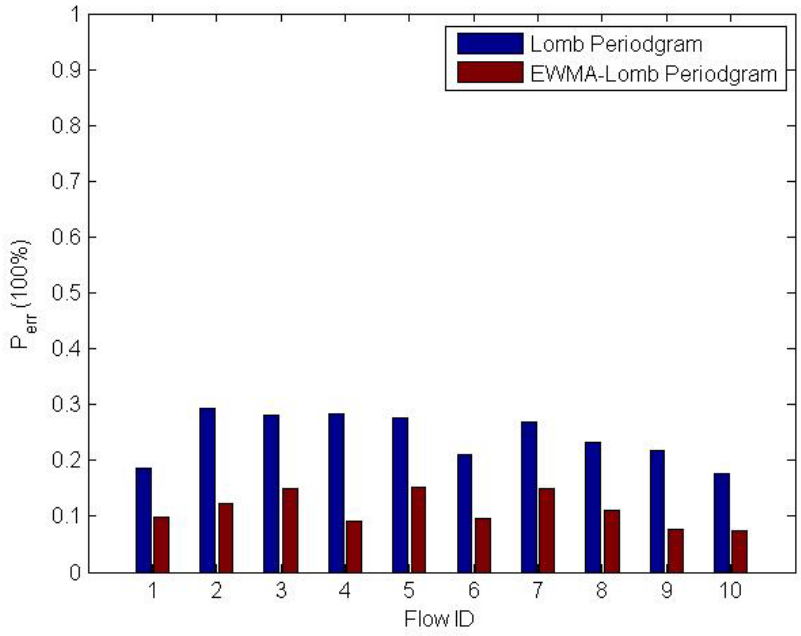


Figure 29: $\mathbf{P_{err}}$ for 10 of 50 simulated TCP flows.

CHAPTER V

DETECTION OF PACKET FLOW ANOMALY (BEHAVIOR IDENTIFICATION)

5.1 Non-deterministic Periods for Cwnd Adjustments

If RTTs of a TCP flow $\mathbf{f_i}$ are estimated using the EWMA Lomb periodogram, then, in theory, its corresponding $\Delta\mathbf{cwnd(t)}$ can be inferred by counting the number of arrival packets during $\mathbf{RTT_i}$. In realty, packets can be lost and retransmitted in any RTT. Additionally, there exist RTT estimation errors and unaligned RTT boundaries. There are two types of errors that can be regarded as the offset error and the scale error. These two errors are sometimes correlated because the accumulated scale error will also lead to the offset error.

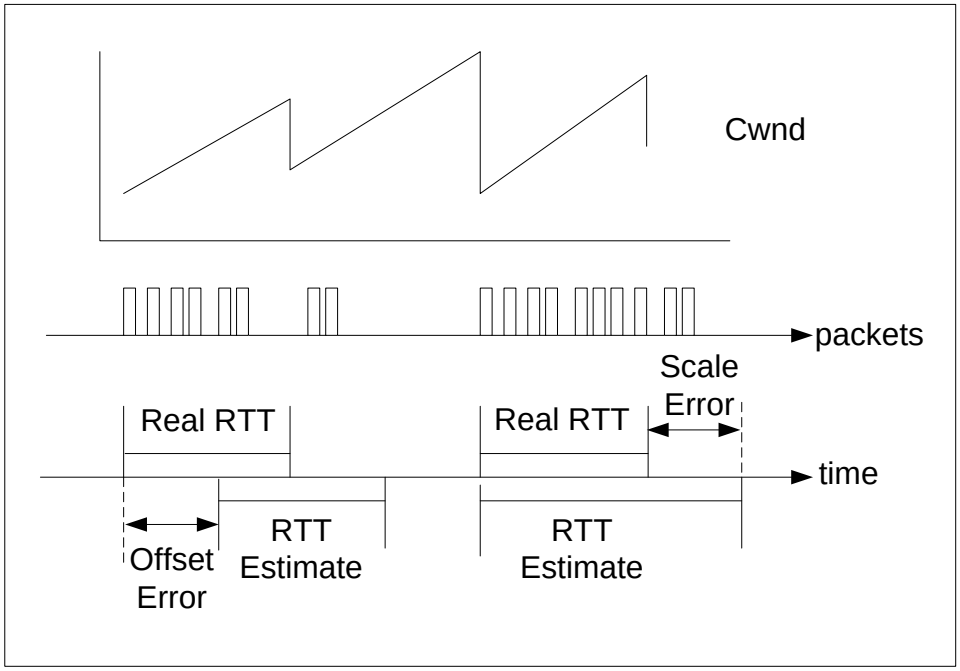


Figure 30: Two types of cwnd estimation errors

Recognizing that the error cannot be completly eliminated, we propose several approaches to decrease the influence of errors when we are detecting $\boldsymbol{\Delta cwnd(t)}$, which is an important parameter to determine the protocol behavior. In this chapter, all possible behaviors are classified into one of following states, i.e. constant rate (CR), zero rate (ZR), additive increase (AI), additive decrease (AD), multiplicative increase (MI), multiplicative decrease (MD), and unknown. In fact, a TCP protocol may not only show one behavior, but different behaviors according to the network condition, e.g. TCP-Reno utilizes the AIMD in the congestion avoidance stage, while MI in the slow start phase. The goal is to detect the unknown TCP protocols if a specific behavior pattern is given as the ground truth, e.g. if TCP-Reno is used as the ground truth, then all non TCP-Reno flows are regarded as unknown.

5.2 Low Pass Filter Approach to Reduce Estimation Errors

Given a sequence of RTT estimates $\{\hat{\theta}_j, j=1,2,3,\ldots\}$ and that of IPT time stamps $\{\tau_k, k=1,2,3,\ldots\}$, where τ_k is the time stamp corresponding to IPT_k, then the estimate of the j-th congestion window

$$\boldsymbol{\xi_j = k - \xi_{j-1}, \xi_0 = 0}\,, \tag{29}$$

where $\boldsymbol{k = \mathbf{argmax}(\tau_k) < \sum_{i=1}^{j} \hat{\theta}_i}$.

The error of the change of the j-th congestion window estimate $\boldsymbol{\Delta\xi_j = \xi_j - \xi_{j-1}}$, is strongly related to $\boldsymbol{\hat{\theta}_j}$ and $\boldsymbol{\hat{\theta}_{j-1}}$. Let $\boldsymbol{\theta_j}$ denote the real RTT and follow the lognormal distribution, i.e. $\boldsymbol{\theta_j \sim LogN(\mu, \sigma^2)}$. Assume that $\boldsymbol{\hat{\theta}}$ is not biased significantly to the

expectation of a set of real RTTs, i.e. $\boldsymbol{E[\widehat{\theta}] - e^{\mu+\sigma^2} \approx 0}$, in a short time interval [$\sum_{i=1}^{j} \widehat{\theta}_i$, $\sum_{i=1}^{j+N} \widehat{\theta}_i$]. Let $\overline{\Delta\xi}(j,N) = \sum_{i=j}^{j+N} \frac{\Delta\xi_i}{N} = (\xi_{j+N-1} - \xi_{j-1})/N$ denote the mean of N adjacent estimates of the windows change in [t_j, t_{j+N}], where $t_j = \sum_{i=1}^{j} \widehat{\theta}_i$ and $N \geq 2$. We also denote $\Delta cwnd(t_j) = cwnd(t_j) - cwnd(t_{j-1})$ the change of the congestion window in two RTTs and $\overline{\Delta cwnd}(j,N) = \sum_{i=j}^{j+N} \frac{\Delta cwnd(t_i)}{N}$ the average change of congestion windows in N RTTs. Assuming that the flow is in the additive increase or decrease state, the variance of the error between $\overline{\Delta\xi}(j,N)$ and $\overline{\Delta cwnd}(j,N)$ will be less than that between $\Delta\xi_j$ and $\Delta cwnd(t_j)$ as proved as follows.

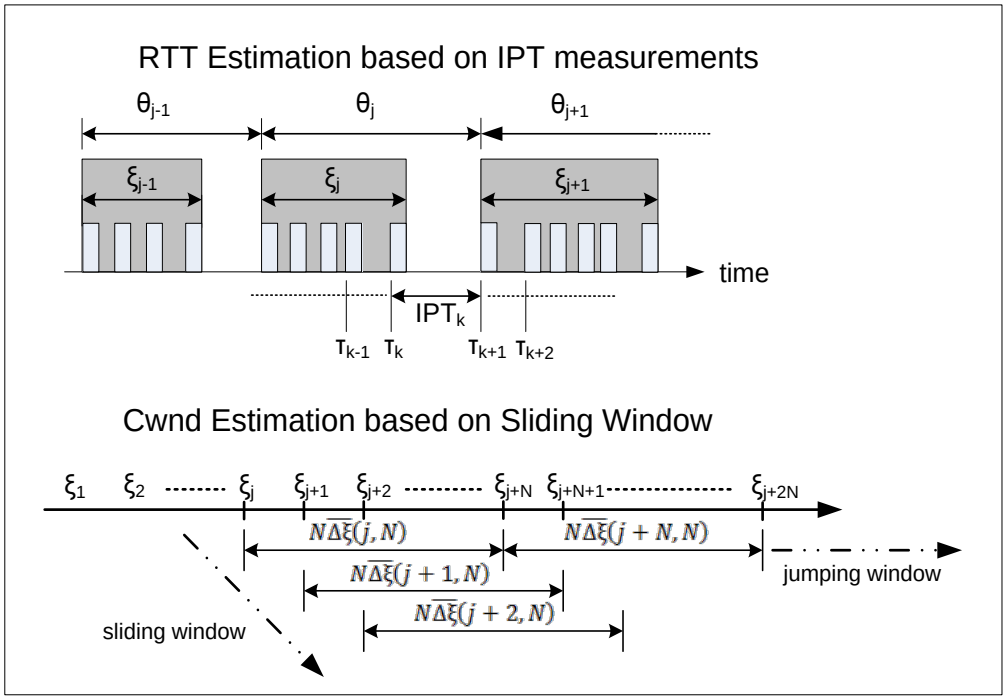


Figure 31: Cwnd estimation based on sliding window

Let $\boldsymbol{\epsilon_j = \Delta\xi_j - \Delta cwnd(t_j)}$ denote the error of the j-th window change estimate, we further define

$$\bar{\epsilon}_{j,N} = \frac{\sum_{i=j}^{j+N}(\Delta\xi_i - \Delta cwnd(t_i))}{N}. \quad (30)$$

Assume that $\boldsymbol{\epsilon = \{\epsilon_1, \epsilon_2, \dots, \epsilon_j, \dots\}}$ is the i.i.d process and follows the normal distribution $\boldsymbol{N(0, \sigma_\varepsilon^2)}$, where $\boldsymbol{\sigma_\varepsilon^2}$ is finite, then $\boldsymbol{E[\bar{\epsilon}_{j,N}] = \frac{\sum_{i=j}^{j+N}(E[\Delta\xi_i] - E[\Delta cwnd(t_i)])}{N} = 0}$ and

$$var[\bar{\epsilon}_{j,N}] = \left(\frac{1}{N}\right)^2 var\left(\sum_{i=j}^{j+N}\varepsilon_i\right) = \frac{\sigma_\varepsilon^2}{N}, \quad (31)$$

Equation (31) indicates that the probability distribution of $\boldsymbol{\bar{\epsilon}_{j,N}}$ is more concentrated around its mean than $\boldsymbol{\varepsilon_j}$ and the variance of $\boldsymbol{\bar{\epsilon}_{j,N}}$ can be further lower by increasing N.

5.3 Dilemma between Detection Sensitivity and Robustness

The sliding window method is suitable for the detection of the long term steady behavior such as AI state, thus minimizing the influence of noises. On the other hand, it may be unable to detect the short-term and abrupt change signals such as the MD state. For the multiplicative change, its duration has to not be long for maintaining the proper network stability and network utilization.

If the behavior of the multiplicative change does not occur consecutively, there will be no sufficient samples to determine multiplicative change by two adjacent congestion windows. Since multiple behavior states may exist in a TCP flow, employing the sliding window to an estimated cwnd sequence will smooth the change of the

congestion window and thus decrease the probability due to the property of the low-pass filter, which is good at detecting the slow change and stable signals.

Instead of analyzing the flow behavior based on a single smoothed curve, multiple state detection modules are proposed to execute simultaneously to provide data for a decision-making algorithm to identify the current state. For each state, a dedicated detection module is used to track if an anomaly occurs for this state. Then the detection outcomes of all detection modules are aggregated and sent to the decision maker such that not only the detection capability can be increased based on the competitive scheme, but also the noise can be reduced if each state detector also behaves like a low-pass filter.

To consider both detection sensitivity and stability, a system consisting of a set of CUSUM banks is proposed to detect the state transition of the flow behaviors.

5.4 Concurrent CUSUM Banks for Cwnd Behavior Detection

Let $\mathbf{M_k}$ denote the behavior metric indexed by k for the N-jumping window sampling, $\mathbf{M_k} = \mathbf{f}(\boldsymbol{\xi}_{\mathbf{k+N-1}}, \boldsymbol{\xi}_{\mathbf{k-1}})$. For $\mathbf{M_k}$, it has an expectation value $\boldsymbol{\eta}$ to determine if a flow is in a specific behavior state. Consider to detect an AI behavior, we have $\mathbf{M_k} = \overline{\Delta\boldsymbol{\xi}}(\mathbf{j}, \mathbf{N}) = (\boldsymbol{\xi}_{\mathbf{k+N-1}} - \boldsymbol{\xi}_{\mathbf{k-1}})/\mathbf{N}$ and $\boldsymbol{\eta} = \mathbf{1}$ ideally.

As the sampling window moves, a series of $\mathbf{M_k}$ can be obtained as $\{\mathbf{M_1}, \mathbf{M_2}, \mathbf{M_3}, \dots, \mathbf{M_\tau}\dots\}$. Our goal is to find a time point $\boldsymbol{\tau}$ such that a flow can be determined to be or not to be in the state X. To detect the change point of $\{\mathbf{M_k}\}$, the cumulative sum (CUSUM) [58][59][60] algorithm will be employed online. We first

assume the flow is in the behavior state X and define $\mathbf{M_k}$ of state X. If the change point occurs, we say the flow is not in state X. However, we are unable to say the flow is in state X if no change point is detected. Therefore, CUSUM here is used to reject the behavior state hypothesis rather than recognizing the initial assumption.

Let $\{\mathbf{M_k}, \mathbf{k} = \mathbf{0}, \mathbf{1}, \mathbf{2} \cdots\}$ denote the sequence of behavior measures at time 0, 1, 2… then we define

$$\mathbf{X_k} = \mathbf{M_k} - \boldsymbol{\eta}, \tag{32}$$

as the behavior change measurement. Here $\{\mathbf{X_k}\}$ is considered as the stationary random process since it is independent of the sampling period and size of the sampling subset. To make CUSUM asymptotically optimal for a wide range of Change Point Detection problems, $\{\mathbf{X_k}\}$ has to satisfy the following two conditions [59]:

(1) $\{\mathbf{X_k}\}$ is $\boldsymbol{\psi}$-mixing, meaning that

$$\boldsymbol{\psi}(\mathbf{s}) \stackrel{\text{def}}{=} \sup_{\mathbf{t} \geq \mathbf{1}} \sup_{\substack{\mathbf{A} \in \mathbf{F_1^t}, \mathbf{B} \in \mathbf{F_{t+s}^\infty} \\ \mathbf{p(A)p(B)} \neq \mathbf{0}}} \left| \frac{\mathbf{P(AB)}}{\mathbf{P(A)P(B)}} - \mathbf{1} \right|,$$

where $\mathbf{F_1^t}$ is the $\boldsymbol{\sigma}$-algebra generated by $\{\mathbf{X_1}, \mathbf{X_2}, \cdots \mathbf{X_t}\}$ and $\mathbf{F_{t+s}^\infty}$ is the $\boldsymbol{\sigma}$-algebra generated by$\{\mathbf{X_{t+s}}, \mathbf{X_{t+s+1}}, \cdots\}$. $\boldsymbol{\psi}(\mathbf{s})$ approaches 0 as $\mathbf{s} \to \infty$, and

(2) The marginal distribution of $\{\mathbf{X_k}\}$ satisfies $\mathbf{E}(\mathbf{e^{tX_k}}) < \infty$, $\exists \mathbf{t} > 0$.

In practice, the above two conditions are very mild and easily satisfied [60] even by long range dependent arrival processes. In the normal condition, we have

$$\mathbf{E}(\mathbf{X_k}) = \boldsymbol{c} \ll \mathbf{1}, \tag{33}$$

i.e. $\boldsymbol{c}$ is much less than 1 and close to 0. Given $\boldsymbol{a}$ the upperbound of $\boldsymbol{c}$, let us denote

$$\widetilde{X_n} = X_n - a, \tag{34}$$

then $\widetilde{X_n}$ will be slightly negative during the normal state and increase significantly when an abrupt change takes place.

If the increase of the mean of $\widetilde{\mathbf{X}_\mathbf{n}}$ is lower-bounded by $\mathbf{h}$ in a specific state, the algorithm of detecting the change from the specific state to another state is described as follows

Let $\mathbf{S_n}$ denote the cumulative sum of $\widetilde{\mathbf{X}_\mathbf{n}}$ such that

$$\mathbf{S_n} = \sum_{i=1}^{n} \widetilde{X_\iota} \text{ with } \mathbf{S_0} = \mathbf{0}, \tag{35}$$

we can denote $\boldsymbol{\mu_n}$ the maximum continuous increment of $\boldsymbol{S_n}$ until time $\boldsymbol{n}$ and then obtain

$$\boldsymbol{\mu_n} = \boldsymbol{S_n} - \min_{\mathbf{1 \le k \le n}} \mathbf{S_k}. \tag{36}$$

It turns out that $\boldsymbol{\mu_n}$ can be expressed as

$$\begin{cases} \boldsymbol{\mu_n} = \left(\boldsymbol{\mu_{n-1}} + \widetilde{X_n}\right)^+ \\ \boldsymbol{\mu_0} = \mathbf{0} \end{cases}. \tag{37}$$

A large $\{\boldsymbol{\mu_n}\}$ is a strong indication of the non expected behavior. Since Equation (37) is recurrent and much easier to compute than Equation (36), it will be used to make detection decisions.

Therefore, given a threshold $\mathbf{N}$, a decision function $\mathbf{d_N}(\boldsymbol{\mu_n})$ can be used to detect the state change such that '0 ' means the original state and '1' the change of state occurs as follows

$$\mathbf{d_N}(\boldsymbol{\mu_n}) = \begin{cases} \mathbf{0 \text{ if } \mu_n} < \boldsymbol{N} \\ \mathbf{1 \text{ if } \mu_n} \ge \boldsymbol{N} \end{cases}. \tag{38}$$

Since the detection time is dependent on the threshold $\mathbf{N}$, we are also interested in their relationships. Let $\boldsymbol{\rho_N}$ denote the normalized detection time and $\mathbf{h}$ the actual increase in the mean of $\{\widetilde{\mathbf{X_n}}\}$ in an anomaly, we have

$$\boldsymbol{\rho_N} = \frac{(\boldsymbol{\tau_N} - \mathbf{w})^+}{\mathbf{N}} \to \frac{\mathbf{1}}{\mathbf{h} - |\mathbf{c} - \mathbf{a}|}, \tag{39}$$

where $\boldsymbol{\tau_N} = \mathbf{inf}\{\mathbf{n}: \mathbf{d_N}(.) = \mathbf{1}\}$, $\mathbf{w}$ is the time index that the abrupt change actually occurs. To ensure the longest time without false alarm and make detection independent of network traffic pattern, $\mathbf{h}$ is suggested to be $\mathbf{2a}$ [60]. However, since $\mathbf{h}$ is a bound rather than a true value, the above is a conservative estimation (upper bound) of the actual detection time.

If AI is set as the normal state, the significant positive change in CUSUM implies the burst size is increasing when the state changes from AI to non-AI. It holds for the $\mathbf{AI} \to \mathbf{MI}$ transition and $\mathbf{AI} \to \mathbf{CR}$ constant rate transition, while may not be true for non-AI states whose burst size is decreased, e.g. $\mathbf{AI} \to \mathbf{AD}, \mathbf{AI} \to \mathbf{MD}$, and $\mathbf{AI} \to \mathbf{CR}$. To detect the abrupt negative change, the behavior detector is required expanded from one-way to two-way. Let $\widehat{\mathbf{X}}_\mathbf{m} = \mathbf{X_m} + \mathbf{a}$ and

$$\hat{\mathbf{S}}_\mathbf{k} = \sum_{i=\boldsymbol{\tau_N}}^{k} \widehat{\mathbf{X}}_\mathbf{i}, \tag{40}$$

i-th $\hat{\mathbf{S}}_{\boldsymbol{\tau_N}} = \mathbf{0}$ and $\boldsymbol{\nu_n}$ denote the maximum continuous decrement of $\hat{\mathbf{S}}_\mathbf{n}$ until time $\boldsymbol{n}$, then we have

$$\boldsymbol{\nu_m} = \hat{\mathbf{S}}_\mathbf{m} - \max_{\boldsymbol{\tau_N} \le \mathbf{k} \le \mathbf{m}} \hat{\mathbf{S}}_\mathbf{k}, \tag{41}$$

and

$$\begin{cases} \boldsymbol{\nu_m} = \left(\boldsymbol{\nu_{m-1}} + \widehat{\mathbf{X}}_\mathbf{m}\right)^- \\ \boldsymbol{\nu_{\tau_N}} = \mathbf{0} \end{cases}. \tag{42}$$

Therefore, given a negative threshold $\widehat{\mathbf{N}}$, a decision function $\mathbf{d_N}(\boldsymbol{\nu_m})$ can be used to detect the negative change such that '0' means the original state is unchanged and '1' the change of state occurs as follows

$$\mathbf{d_N}(\boldsymbol{\nu_m}) = \begin{cases} \mathbf{0 \text{ if } \nu_m} > \widehat{\mathbf{N}} \\ \mathbf{1 \text{ if } \nu_m} \leq \widehat{\mathbf{N}} \end{cases} \tag{43}$$

Table 4 lists possible behavior states of a flow. A traditional TCP-Reno protocol has the MI state for the slow start phase, AI state for the congestion avoidance phase and MD state corresponding to packet loss. Opposite to AI state, we have the AD, which exists in TCP-Vegas to decrease its rate if the bottleneck queue length exceeds a threshold. CR is a major state of Vegas so that the TCP-Vegas flow can keep a constant rate for a longer time. Compared to CR, ZR means the zero rates, which can occur in some on-off flows. For flows neither showing the strong multiplicative change nor the additive change, their states are regarded as unknown, which means flows are affected by strong noises in networks.

Table 4: State definition

State	Definition
AI	Additive Increase
AD	Additive Decrease
MI	Multiplicative Increase
MD	Multiplicative Decrease
CR	Constant Rate
ZR	Zero Rate
Unknown	Unknown State

AIMD is a typical behavior of TCP flows. Recall that $\boldsymbol{\xi_i}$ represents the i-th estimated cwnd, the equation $\boldsymbol{\xi_k - \xi_{k-1} = 1}$ holds for the AI behavior if no noise is introduced to the measurement. Therefore the linear gradient of $\boldsymbol{\xi_k}$, e.g. $\boldsymbol{\xi_k - \xi_{k-1}}$, is a good behavior metric for AI. The same reason is also applied to AD. Hence the behavior metric $\boldsymbol{M_k = \xi_k - \xi_{k-1}}$ will be defined as the linear gradient indicator to detect the additive change of cwnd in the CUSUM algorithm.

For the MD behavior, the size of the congestion window doubles every RTT, i.e. the cwnd grows exponentially. The linear gradient is no longer suitable to detect the MI behavior because $\boldsymbol{M_k}$ has to be a stationary random process to meet the requirement of CUSUM. If the multiplicative change occurs at $\boldsymbol{RTT_i}$, then the following equation $\boldsymbol{\frac{\xi_i}{\xi_{i-1}} = k}$ holds, where k is the multiplicative factor. By taking the logarithm of the ratio, we can transform the exponential change into the linear one as shown in Equation (44).

$$\boldsymbol{M_k = \log\left(\frac{\xi_i}{\xi_{j-1}}\right) = \log(\xi_i) - \log(\xi_{i-1}).} \tag{44}$$

The log gradient based on Equation (44) is therefore used as the metric of detecting the multiplicative change of the cwnd.

Different from the linear and log gradient metrics, zero gradients is not based on the difference of two adjacent cwnds, but the cwnd only. For some ON-OFF flows, the zero gradient can be employed to identify the zero rate in the OFF period. Table 5 shows $\boldsymbol{M_k}$ of different gradients expressed in terms of $\boldsymbol{\xi_i}$.

Table 5: Gradient-based behavior metrics

Gradient	Definition
Linear gradient	$\mathbf{M_k = \xi_k - \xi_{k-1}}$
Log gradient	$\mathbf{M_k = \log(\xi_k) - \log(\xi_{k-1})}$
Zero gradient	$\mathbf{M_k = \xi_k}$

Figure 32 shows the flow state is detected based on a set of CUSUM banks. The cwnd estimates are fed into each of the banks simultaneously. Then the decision system will ask CUSUM detectors if they think their predefined state changed. Based on the answers of all CUSUM detectors, the system will determine the current state of the flow.

For AI, $\mathbf{X_k}$ is denoted as $\mathbf{M_k - \eta}$ as shown in Equation (32), where $\mathbf{M_k = \xi_k - \xi_{k-1}}$ and $\boldsymbol{\eta = 1}$ since cwnd increases by 1 after one RTT such that $\mathbf{E(X_k) = c \ll 1}$ holds in Equation (33). In practice, due to network noises, $\mathbf{a}$, the upper bound of $\mathbf{c}$ in Equation (34) is set to 1. Therefore $\widetilde{\mathbf{X_n}} = \mathbf{X_n - a}$ is equivalent to $\widetilde{\mathbf{X_n}} = \mathbf{M_k - \beta}$, where $\boldsymbol{\beta = 2}$, for the AI detection. Since $\mathbf{h = 2a}$ is suggested in [60] and the shortest detection delay is required, then $(\boldsymbol{\tau_N - w})^+$ is set to 1 and the Non-AI CUSM threshold $\mathbf{N}$ in Equation (38) is equal to 1 based on Equation (39).

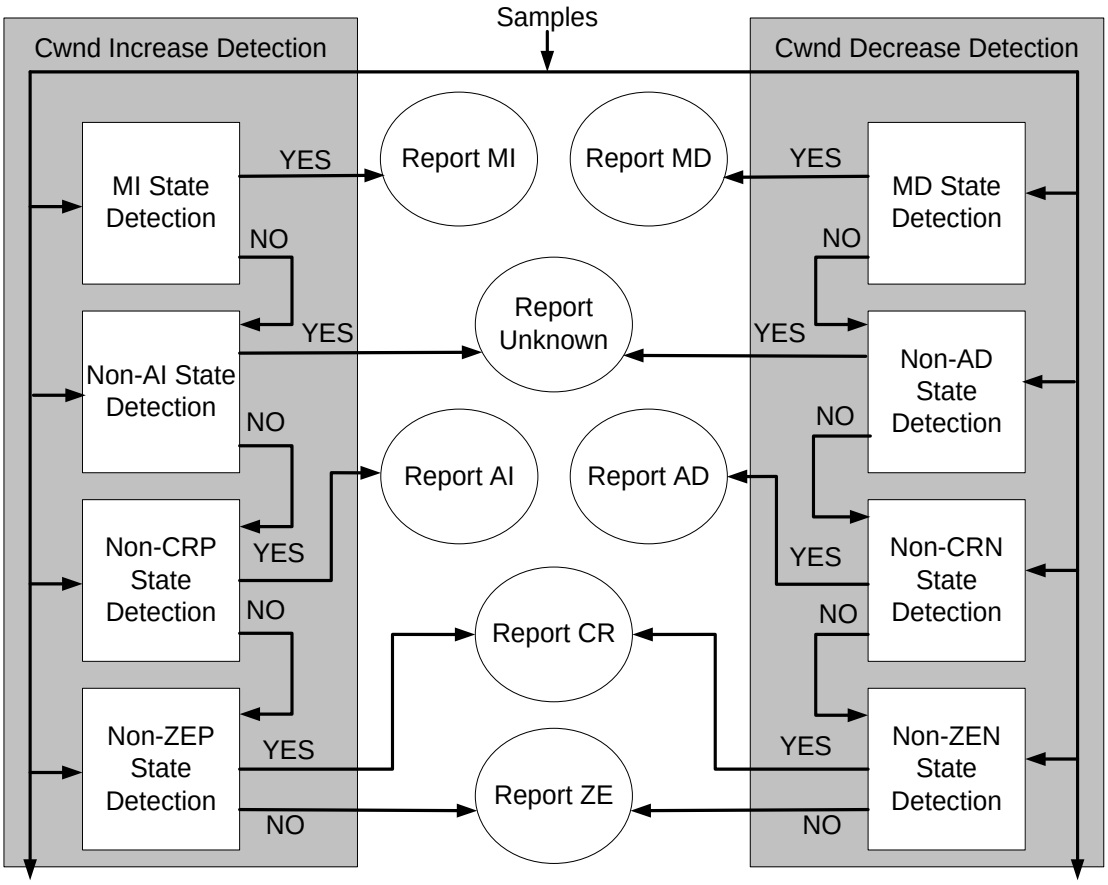


Figure 32: Detection of the flow behavior based on CUSUM banks

To detect CRP, the CR with the positive change of cwnd, we have $\boldsymbol{\eta = 0}$ in Equation (32) and set $\mathbf{a = 1}$ in Equation (34). Then $\widetilde{\mathbf{X_n}} = \mathbf{M_k} - \boldsymbol{\beta}$, where $\mathbf{M_k}$ is the linear gradient metric and $\boldsymbol{\beta = 1}$ such that non-CR occurs if $\boldsymbol{\mu_n > N}$ as shown in Equation (38), where N=1, given the shortest detection delay. For the detection of ZEP, zero rate with the positive change, we have $\mathbf{M_k} = \boldsymbol{\xi_k}$ and $\mathbf{a = 1}$.

Multiplicative change is significant in the congestion window size. Hence, multiplicative changes occur less frequently compared to additive changes. To detect the

MI behavior, different from the AI, CR and ZR detection, the CUSUM detector regards the non-MI the normal state and MI the abnormal state. Once the alarm flags, we say the flow is in the MI state. **Let**

$$\frac{\boldsymbol{\xi_k}}{\boldsymbol{\xi_{k-1}}} \leq \boldsymbol{2(1-p)} \tag{45}$$

indicate a non MI state, where $\mathbf{p}$ is the percentage of the noise tolerance for the MI state. By taking the logarithm of Equation (45), we have $\mathbf{M_k = \log_2(\xi_k) - \log_2\ (\xi_{k-1})}$ with $\boldsymbol{\eta = 1 + \log_2\ (1-p)}$ such that $\mathbf{X_k = M_k - \eta}$. By setting the upperbound $\mathbf{a}$ of mean of $\{\mathbf{X_k}\}$ to 0, then we have $\widetilde{\mathbf{X_n}} = \mathbf{\log_2(\xi_k) - \log_2(\xi_{k-1}) - 1 - \log_2\ (1-p)}$ and N is set to $\mathbf{\log_2(\frac{1-0.5p}{1-p})}$ if $\mathbf{E(M_k) = \log_2(2 * (1-0.5p))}$ is regarded as the occurance of MI, where $\mathbf{p}$ is set to 0.25.

For the detection of the decrease change of the cwnd value, the same logic can be applied to set CUSUM parameters for MD, AD, CRN and ZEN. Table 6 shows behavior metrics for distinct CUSUM banks and Table 7 defines the detection sequence and CUSUM threshold for each CUSUM bank.

Table 6: Behavior metrics of CUSUM detectors

Detector	Behavior Metric
MI	$\mathbf{M_k = \log\ (\xi_k) - \log\ (\xi_{k-1})}$
MD	$\mathbf{M_k = \log\ (\xi_k) - \log\ (\xi_{k-1})}$
Non-AI	$\mathbf{M_k = \xi_k - \xi_{k-1}}$
Non-AD	$\mathbf{M_k = \xi_k - \xi_{k-1}}$
Non-ZEP	$\mathbf{M_k = \xi_k}$
Non-ZEN	$\mathbf{M_k = \xi_k}$
Non-CRP	$\mathbf{M_k = \xi_k - \xi_{k-1}}$
Non-CRN	$\mathbf{M_k = \xi_k - \xi_{k-1}}$

Table 7: Parameters of CUSUM banks

Detector	Detection Sequence	CUSUM Threshold
Non-ZEP	$\widetilde{\mathbf{X_n}} = \mathbf{M_k} - \mathbf{1}$	$\mathbf{N} = \mathbf{1}$
Non-ZEN	$\widetilde{\mathbf{X_n}} = \mathbf{M_k} + \mathbf{1}$	$\mathbf{N} = -\mathbf{1}$
Non-CRP	$\widetilde{\mathbf{X_n}} = \mathbf{M_k} - \mathbf{1}$	$\mathbf{N} = \mathbf{1}$
Non-CRN	$\widetilde{\mathbf{X_n}} = \mathbf{M_k} + \mathbf{1}$	$\mathbf{N} = -\mathbf{1}$
Non-AI	$\widetilde{\mathbf{X_n}} = \mathbf{M_k} - \mathbf{2}$	$\mathbf{N} = \mathbf{1}$
Non-AD	$\widetilde{\mathbf{X_n}} = \mathbf{M_k} + \mathbf{2}$	$\mathbf{N} = -\mathbf{1}$
MI	$\widetilde{\mathbf{X_n}} = \mathbf{M_k} - \mathbf{log_2(1.5)}$	$\mathbf{N} = \mathbf{log_2(1.16)}$
MD	$\widetilde{\mathbf{X_n}} = \mathbf{M_k} + \mathbf{log_2(1.5)}$	$\mathbf{N} = -\mathbf{log_2(1.16)}$

Since the behavior state changes frequently, the CUSUM has to be reset to make the state detection work properly based on following rules:

1. Each CUSUM bank is assigned a priority.
2. Priority orders of increase detection: MI > Non-AI > Non-CRP > Non-ZEP.
3. Priority orders of decrease detection: MD > Non-AD > Non-CRN > Non-ZEN.
4. When a state is detected, all CUSUM banks with the priority less than equal to the detected one are reset to 0.

Again, the operation workflow of the behavior detection system is shown in Figure 32.

5.5 Lossy Finite State Machine of Detecting Unknown Protocols

Each TCP protocol has its own pattern of state transitions. The abnormal state transition is an indication of unknown protocols. Given a 5-tuple deterministic finite automata (DFA) $\mathbf{M} = (\mathbf{Q}, \mathbf{\Sigma}, \boldsymbol{\delta}, \mathbf{q_0}, \mathbf{F})$ is defined as follows:

1. a finite set of states ($\mathbf{Q}$)
2. a finite set of input symbols called the alphabet ($\mathbf{\Sigma}$)
3. a transition function ($\mathbf{\delta} : \mathbf{Q} \times \mathbf{\Sigma} \rightarrow \mathbf{Q}$)
4. a start state ($\mathbf{q_0} \in \mathbf{Q}$)
5. a set of accept states ($\mathbf{F} \subseteq \mathbf{Q}$)

Transitions between states in the finite state machine are based on the RTT estimates and cwnd estimates. The behavior of well-known protocols can be defined by **M** and used as the ground truth to detect other protocols. No estimation can be 100% correct. Packet loss, excessive delay or retransmission, and any other traffic disruptions, may lead to false state identification. To cope with these conditions, additional states to account for these disruptions are added to the congestion control states. In Figure 33, the solid circle at the right side shows the finite state machine of TCP-Reno. The dashed circle contains aberrant states at the left side.

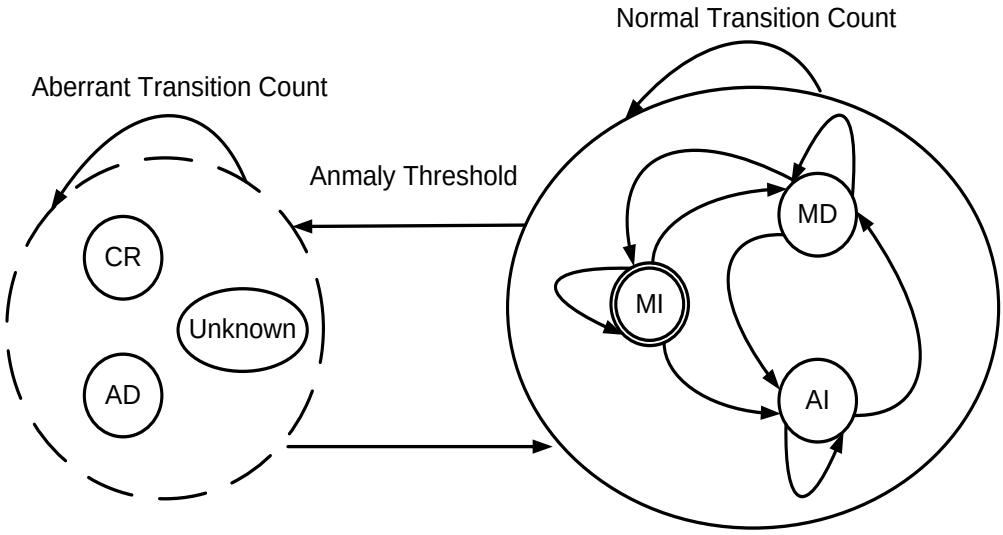


Figure 33: TCP-Reno finite state machine

The aberrant transition count (ATC) will be increased by 1 when an illegal state transition occurs; the error transition will be ignored if ATC is less than the anomaly threshold. The ATC will be reset to 0 if a normal transition occurs in the FSM. When ATC is updated and its value exceeds the anomaly threshold, the aberrant transition count (ATC) will be increased by 1. On the other hand, the normal transition count (NTC) represents how many times the state transition of a flow follows the protocol DFA. NTC is increased by 1 for each well-defined transition.

Given a detection ratio threshold $\mathbf{R_{th}}$ and a specific number N of samples, we say a flow belongs to the TCP protocol defined by the DFA if

$$\mathbf{R} = \frac{\mathbf{ATC}}{\mathbf{NTC}} < \mathbf{R_{th}} \tag{46}$$

based on N samples. Then flows of different protocols will be identified using the R-based pattern.

CHAPTER VI

PERFORMANCE OF PACKET FLOW ANOMALY DETECTION

6.1 Experiment Configuration

To verify the detection rate of the packet flow anomaly detection system, we use NS2 to generate flows and set the dumbbell as the topology as shown in Figure 34. For the link of S_i to O_1, where i is an integer, its bandwidth is 10 Mbps and propagation delay is 10 ms. The bottleneck between O_1 and O_2 has the bandwidth 10 Mbps and transmission delay 20 ms. For links of O_2 to R_j, where j is an arbitrary integer, its bandwidth and propagation delay are the same as the links of S_i to O_1.

Based on the configuration of Figure 34, five different experiments are performed to evaluate the feasibility and performance of the flow anomaly detection system.

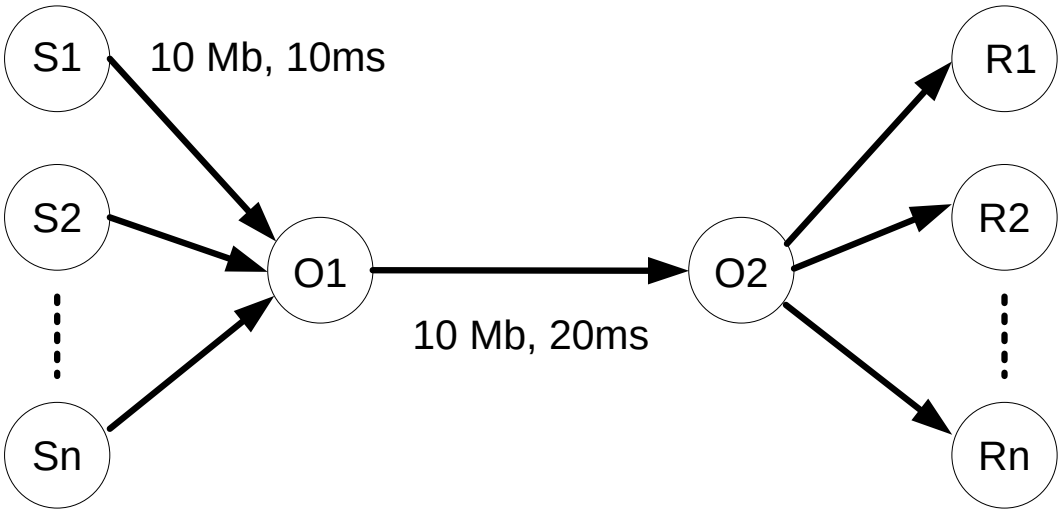


Figure 34: Network topology of experiments

6.2 Performance Evaluation

6.2.1 Cwnd Estimation

Figure 35 shows the dynamic change of cwnd using the approach of Lomb-based packet counting. From the diagram, the proposed cwnd estimation method does capture the characteristic of the slow start and AIMD of TCP-Reno flows.

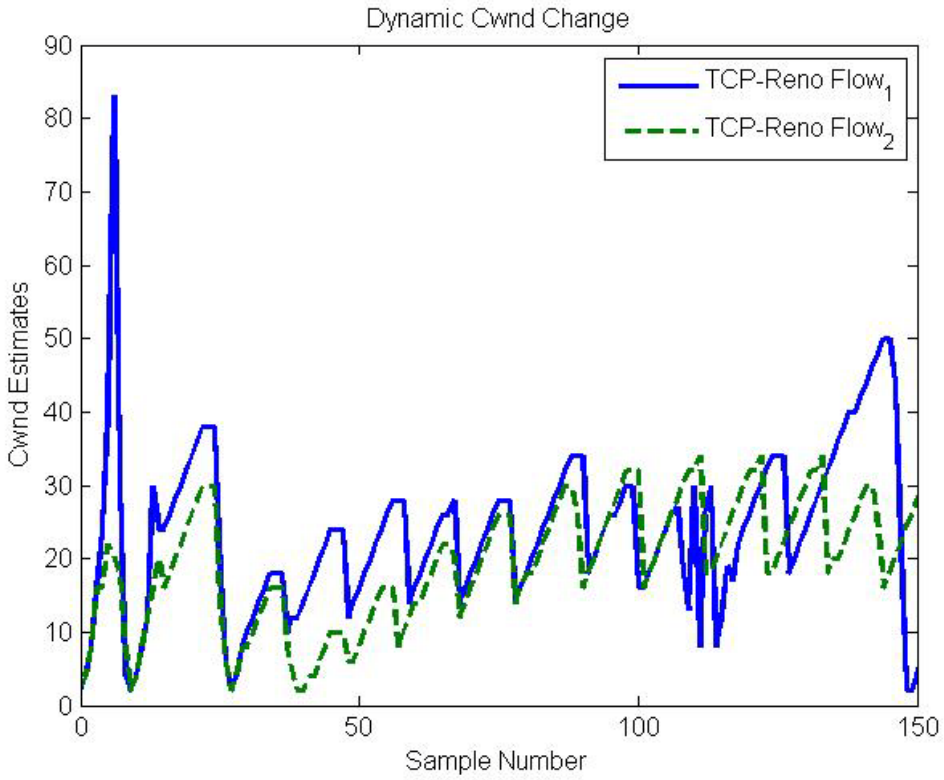


Figure 35: Cwnd estimates of TCP Reno flows

Figure 36 shows the outcome of state detection of one of 10 TCP Reno flows, unknown states are the change of congestion windows between additive change and multiplicative change. Because of the high noise environment, the additive change can

be easily detected as the constant rate. For TCP Reno, AI, MI and MD are three possible states. Figure 37 shows the count of the normal transitions and that of aberrant transitions for each flow. The number of right transitions is higher than its counterpart. Figure 38 represents the ratio of the right transitions to the error transitions for each flow. The ratios of the 10 flows are 0.209, 0.168, 0.129, 0.140, 0.126, 0.171, 0.119, 0.221, 0.172, and 0.114 respectively. Among all these ratios, the maximum one is 0.221. If the ratio threshold is set to be higher than 0.221, then the detection rate will be 100%.

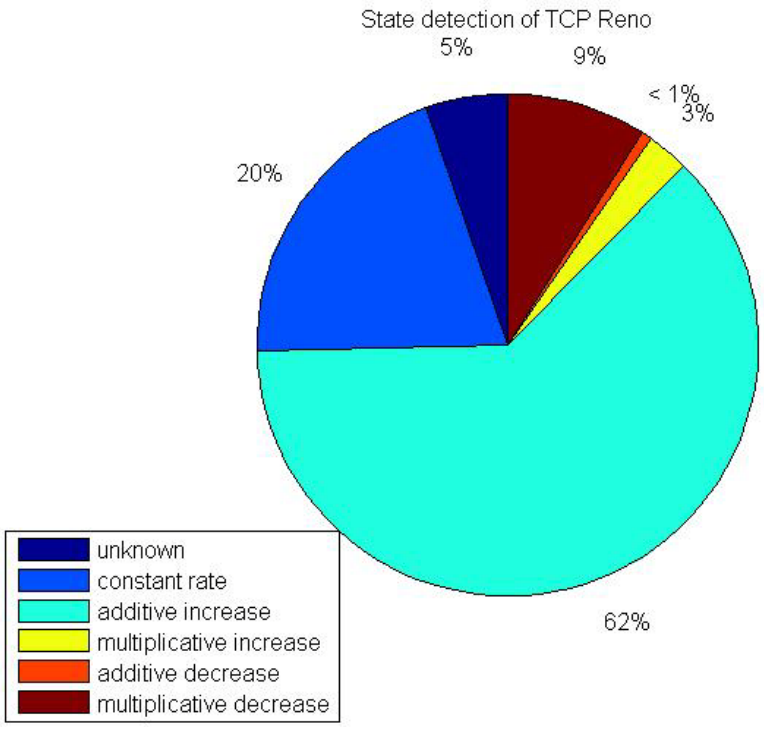


Figure 36: State transition diagram

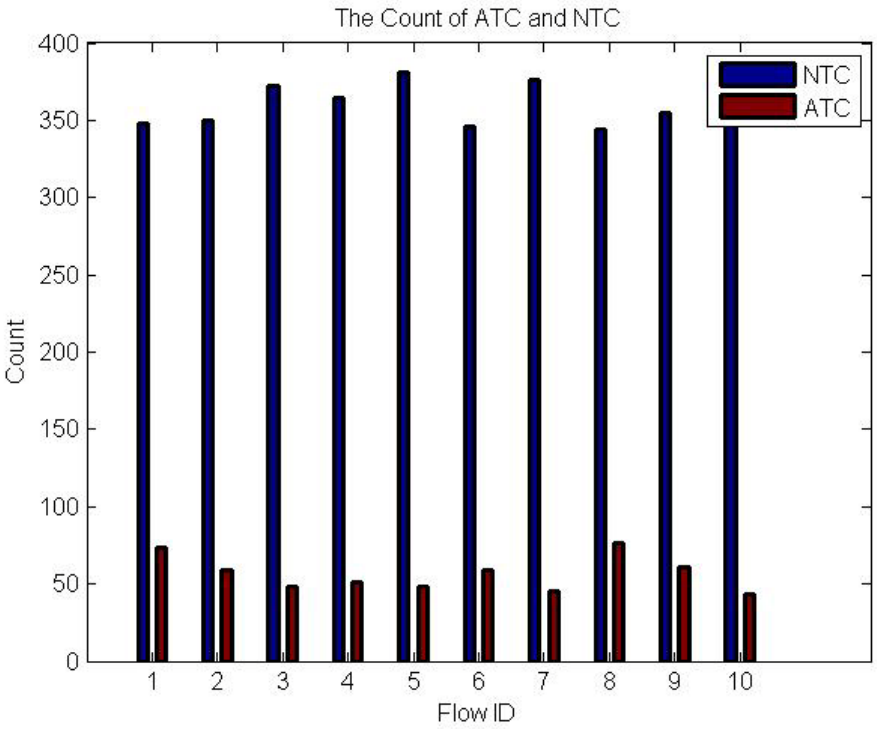


Figure 37: Count of ATC and NTC

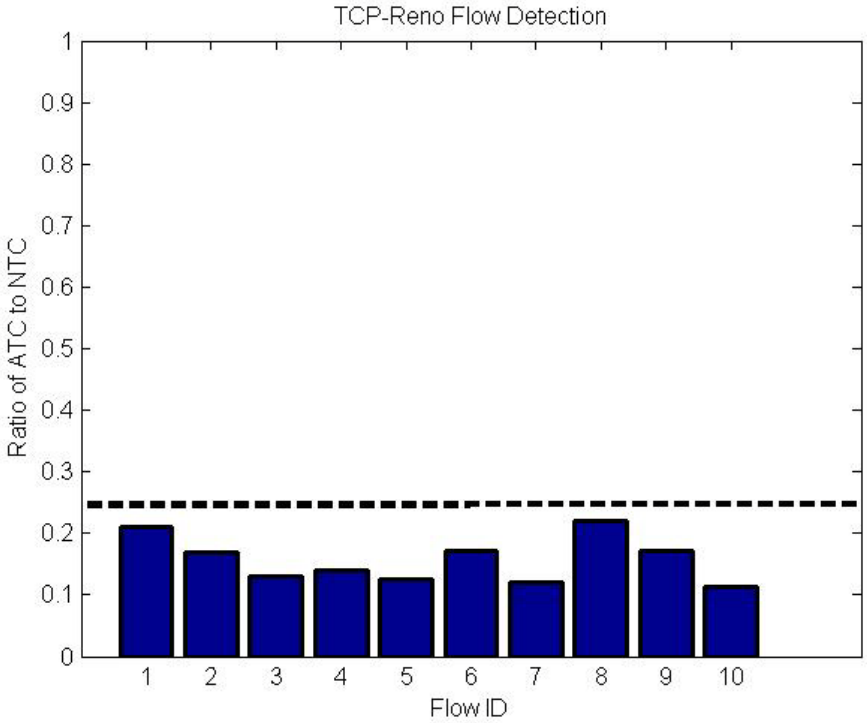


Figure 38: Ratio of ATC to NTC

Because the detection rate is dependent on the ratio threshold, Figure 39 shows the detection rate in different ratio thresholds. Although the higher threshold leads to the higher detection rate, it may also cause more false positives. In the following experiments 2 and 3, we will show its impact to false positives.

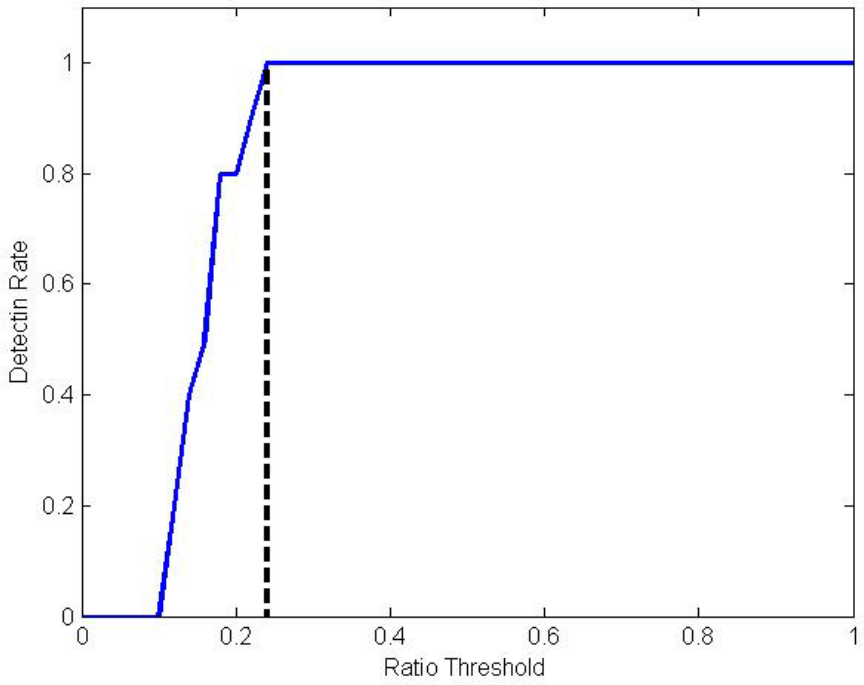


Figure 39: Detection rate versus ratio threshold.

6.2.2 Differentiation of TCP-Reno from CBR

Based on the same topology as shown in experiment one. Experiment two aims to investigate the detection rate and false alarm rate among different flows. Therefore, 5 more CBR, constant bit rate, flows with 1Mb/sec each are added into the simulation.

Since the Lomb-periodogram is applied to compute RTTs and cwnds of each flow, CBR flows also have "virtual" cwnds measured at the access point as shown in Figure 40. Due to network buffer overflow and packet loss, the estimated cwnds of CBR flows are not always constant, but influenced by the network congestion conditions.

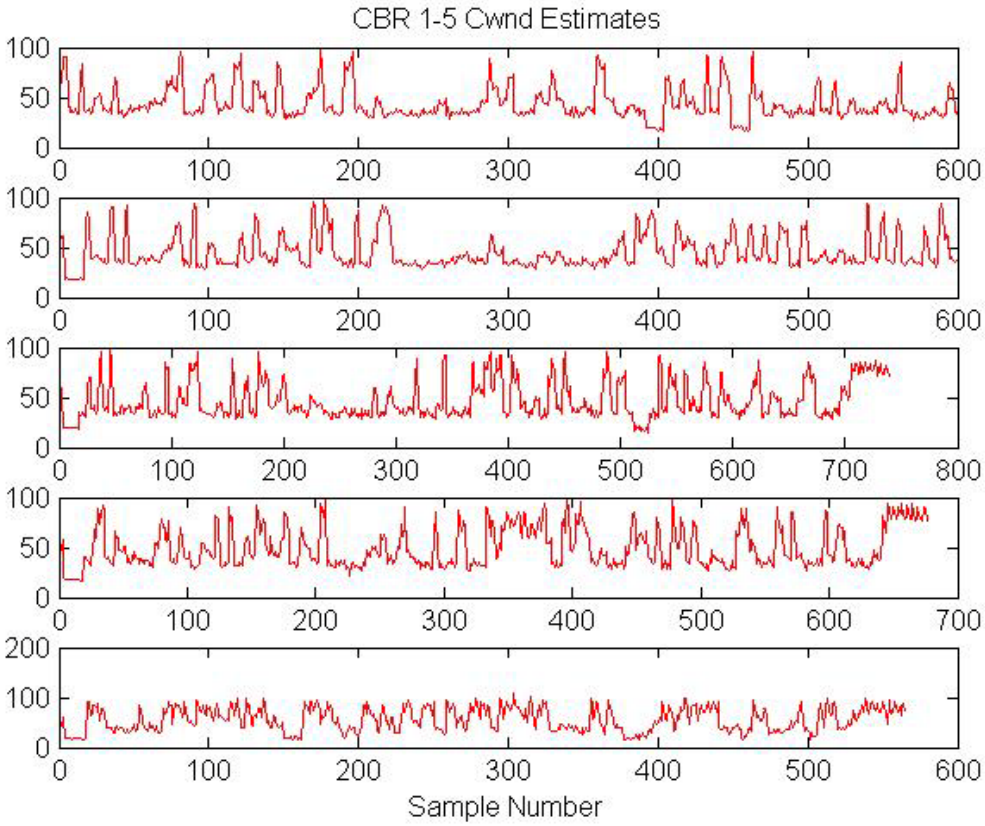


Figure 40: Virtual congestion window estimates of UDP

By feeding the estimated cwnds into the CUSUM-based FSM, the ratio of ETC to RTC is a good indicator to classify TCP-Reno flows and CBR flows into two areas as shown in Figure 41. The ratios of 10 TCP-Reno flows are 0.178, 0.195, 0.234, 0.148,

0.129, 0.227, 0.165, 0.158, 0.181, and 0.209 respectively. On the other hand, the ratios of 5 CBR flows are 7.653, 6.859, 8.458, 9.155 and 10.268 respectively. Therefore, by properly setting the $\mathbf{R_{th}}$ between 0.234 and 6.859, two flow types can be identified correctly.

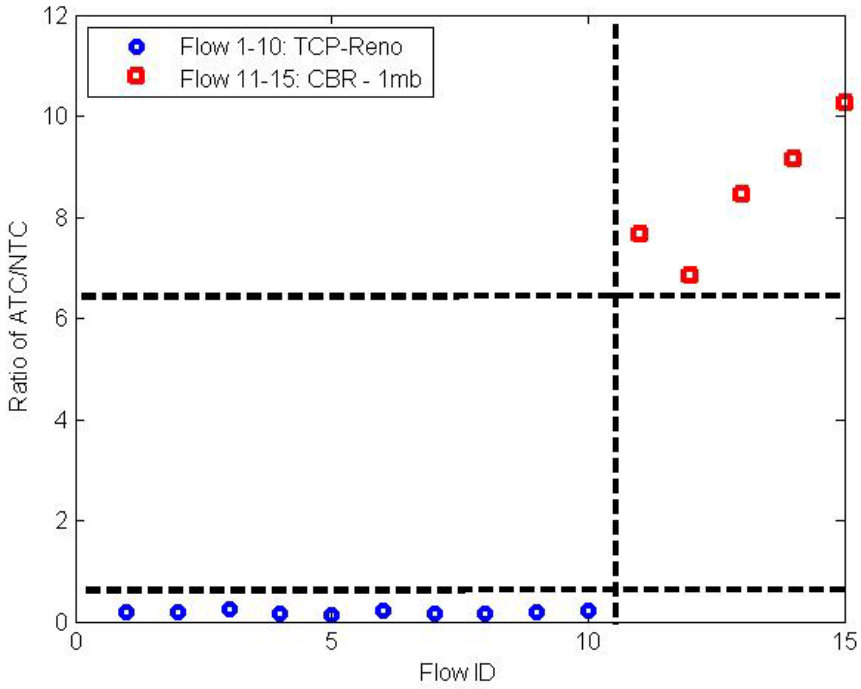


Figure 41: Classification of TCP-Reno and non-Reno flows based on the ratio of ETC to RTC.

6.2.3 Differentiation of TCP-Reno from TCP-Vegas

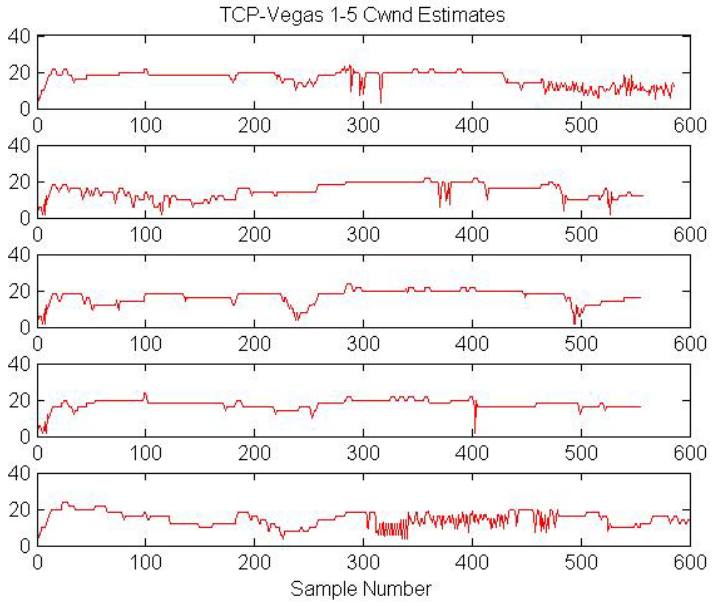
This experiment is the same as experiment two except that 5 TCP-Vegas flows are used instead of 5 CBR flows. In the network area, the two most popular TCP protocols for investigation are TCP-Reno and TCP-Vegas. For Vegas, its congestion control protocol is depicted as follows. In the slow start phase, TCP-Vegas doubles

$\mathbf{cwnd_i}$ every two RTTs to detect and prevent the packet loss. When the network queue length is greater than $\boldsymbol{\gamma}$, Vegas will enter the congestion avoidance phase and reduce $\mathbf{cwnd_i}$ by 1/8. In congestion avoidance, Vegas infers the expected rate as $\frac{\mathbf{cwnd_i}}{\mathbf{BaseRTT}}$, where $\mathbf{BaseRTT}$ is the minimum of all measured RTT, and physical rate as $\frac{\mathbf{transmitted\ bytes}}{\mathbf{measured\ RTT}}$, then it adjusts the window size as follows

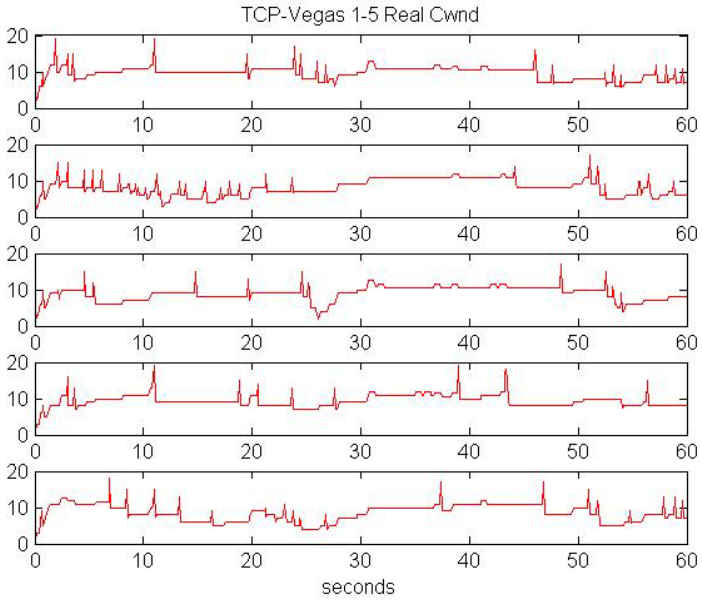
$$\boldsymbol{cwnd_i(n+1)} = \begin{cases} \boldsymbol{cwnd_i(n)+1} & \mathbf{, diff < \alpha} \\ \boldsymbol{cwnd_i(n)} & \mathbf{, \alpha \le diff \le \beta,} \\ \boldsymbol{cwnd_i(n)-1} & \mathbf{, diff > \beta} \end{cases} \tag{47}$$

where $\mathbf{diff} = \frac{\boldsymbol{cwnd_i}}{\mathbf{BaseRTT}} - \frac{\mathbf{transmitted\ bytes}}{\mathbf{measured\ RTT}}$ is an estimate of the number of packets of the flow that are backlogged in the network.

Figure 42 (a) shows that the congestion windows of TCP-Vegas flows tend to stay at a constant level longer compared to the AIMD property of TCP-Reno flows. Therefore TCP-Vegas behaves conservatively corresponding to the network congestion. Figure 42 (b) is the ground truth of congestion windows of 5 TCP Vegas flows. Figure 43 shows TCP-Reno flows and TCP-Vegas flows can be properly distinguished if R_{th} is set between 0.262 and 5.714 given that the anomaly threshold is set to 1.



(a)



(b)

Figure 42: Congestion windows of 5 TCP-Vegas flows: (a) estimated states, and (b) actual states.

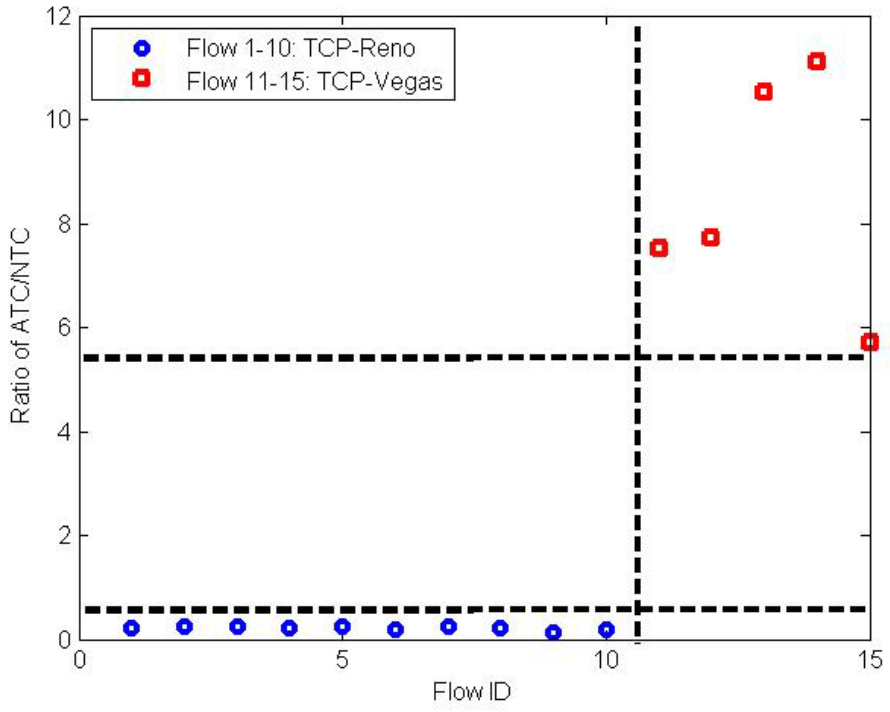


Figure 43: Classification of two different TCP flavors

6.2.4 Classification of Mixing Flows Using TCP-Reno FSM

To study the influence of different flows on the detection rate, 50 flows consisting of 15 TCP-Reno flows, 15 TCP-Vegas flows, 10 CBR flows of 1 Mb/sec and 10 CBR flows of 2 Mb/sec are sent into the dumbbell topology, where the bandwidth of the o1-o2 link is set to 50 Mb/sec and other configurations are unchanged.

The goal of this experiment is to verify if the TCP-Reno FSM is able to distinguish TCP-Reno flows from others. Figure 44 shows the estimated cwnd size for flows of different protocols. In the diagram, TCP-Reno shows the saw-tooth shape; TCP-Vegas is adaptive to the congestion condition using AI and AD and tends to have an unchanged cwnd if the network traffic does not change significantly. In addition,

CBR flows have the stronger constant rate property compared to TCP-Vegas because they are not responding to the traffic congestion.

Figure 45 shows that the ratio of ATC to NTC is between 0.14 and 0.59 for TCP-Reno flows, between 0.79 and 4.13 for TCP-Vegas, between 5.95 and 15.24 for 1MBps CBR flows and between 4.95 and 8.74 for 2Mbps CBR flows. Therefore, TCP-Reno flows and Non-Reno ones can be distinguished if $\mathbf{R_{th}}$ is properly set between 0.59 and 0.79. Table 8 lists the mean and variance of R of 50 flows. That being said, it is difficult to differentiate the two classes when their R values have much smaller differences than in earlier cases due to the sharply diminished mean and variance values. This situation was caused by significant presence of CBR flows, which significantly increases congestion conditions, so that estimation of the RTT for both Reno and Vegas flows tends to be disrupted, which then translates into erroneous predictions of congestion control states. The gradually converged behaviors between Reno and Vegas can be visualized in Figure 44, where the sharply distinct dynamics of their congestion window changes diminished.

Table 8: Mean and variance of ratio of ATC to NTC

Flow ID	Protocol	Mean	Variance
1-15	Reno	0.264	0.011
16-30	Vegas	1.109	0.709
31-40	CBR(1Mbps)	11.609	10.681
41-50	CBR(2Mbps)	6.644	1.6246

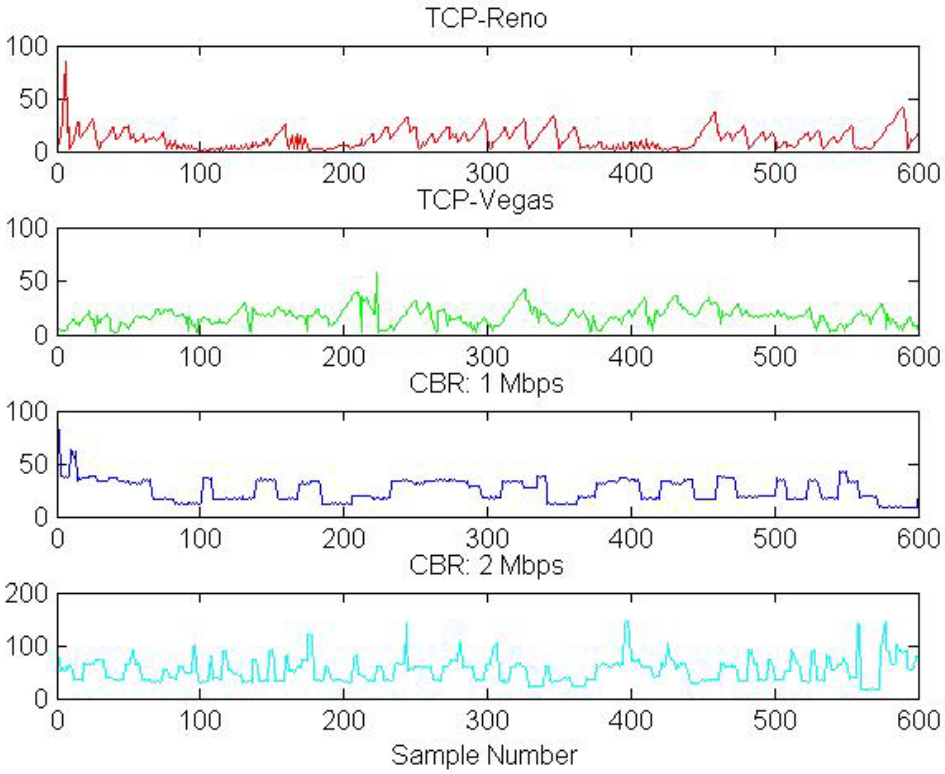


Figure 44: Estimates of congestion windows.

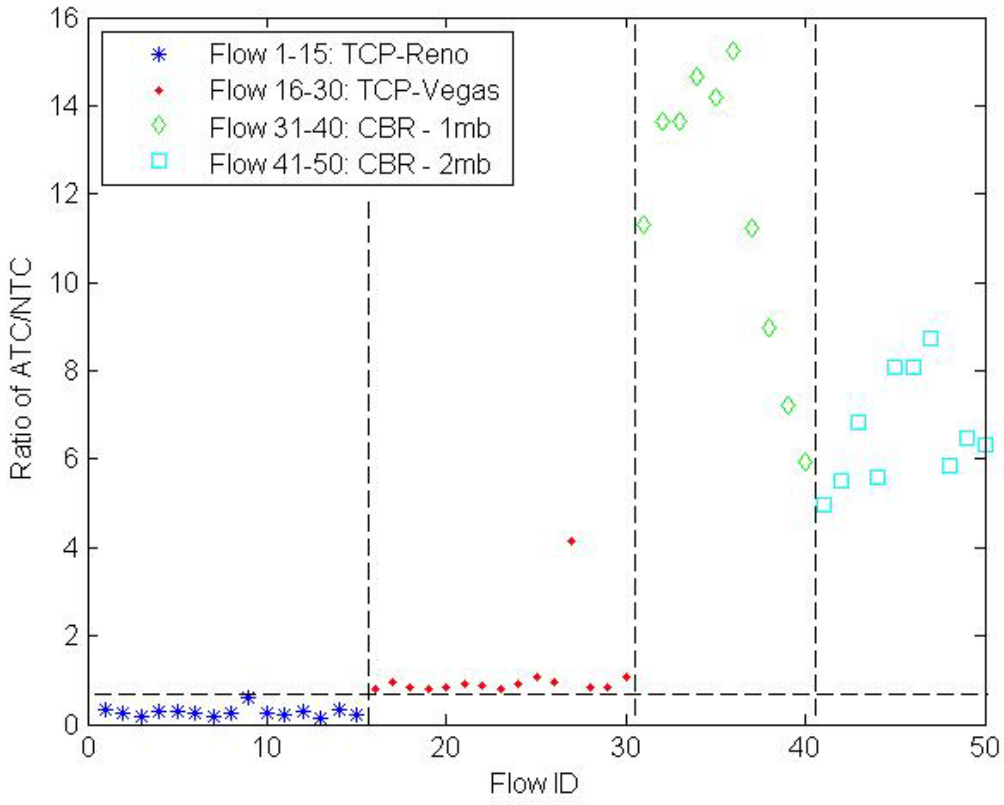


Figure 45: Classification in mixed flow types

6.2.5 Classification of Mixing Flows Using TCP-Vegas FSM

In this experiment, we show that a FSM based on TCP-Vegas can detect non-TCP-Vegas flows with 100% detection rate and one false alarm among 50 flows.

Figure 46 shows the state transitions of flows of three different protocols. Compared with Reno, Vegas has a lower transition frequency and prefers to stay at the CR state (state 1). In addition, Vegas is a conservative protocol so that its rate adjustment is not drastic and therefore the probability it enters the MD state (state 6) is

lower than that of Reno. CBR flows stay longer in the CR state as expected because CBR flows are not controlled by the congestion protocol and only influenced by the queue delay and overflow. For CBR flows, the one with the higher rate has a more frequent state transition than that with the lower rate. This is because the higher rate CBR has more packets in the network, thus increasing the probability of packet loss and delay, which will respond to the state change later.

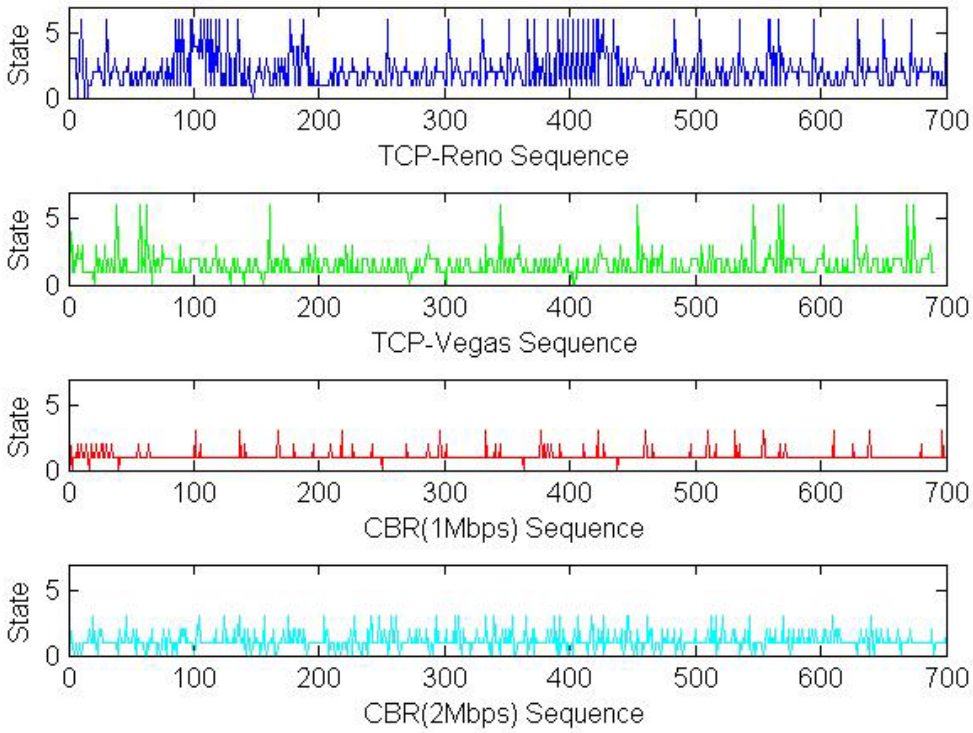


Figure 46: Cwnd dynamics of different flow types

Since the major property of TCP-Vegas is the constant rate, we construct a simple FSM, as shown in Table 9 based on this major feature and observe the

performance of its detection outcome. Figure 47 shows the ratio of 50 flows plotted in Figure 48.

Table 9: Ratio of ATC to NTC for 50 flows

Flow ID	Protocol	Ratio
1-15	Reno	0.74-1.26
16-30	Vegas	0.39-0.54 with an outlier 0.11
31-40	CBR(1Mbps)	0.002-0.073
41-50	CBR(2Mbps)	0.085-0.241

The CR FSM can distinguish Reno from Vegas with 100% detection rate if $\mathbf{R_{th}}$ is set between 0.54 and 0.74. The mean of R of 15 Reno flows is 1.034 and its corresponding variance is 0.031. On the other hand, the mean of R of 15 Vegas flows is 0.430 and its corresponding variance is 0.009.

Although Reno and Vegas can be distinguished, one $\mathbf{R_{th}}$ cannot classify Vegas and CBR because the R of CBR flows is lower than that of TCP-Vegas flows. The reason is that Vegas is adaptive to the network congestion and therefore cannot stay at the CR state as long as CBR flows do. However, if another $\mathbf{R_{th}}$ regarded as the lower-bound threshold is set between 0.54 and 0.241, then Vegas flows and CBR flows can be classified into two groups with only one false alarm, which is an outlier of Vegas flows, among 50 simulated flows in the noisy and dynamic networks. Furthermore, CBR flows with two different rates are able to be classified with 100% correct rate if $\mathbf{R_{th}}$ is set between 0.073 and 0.085.

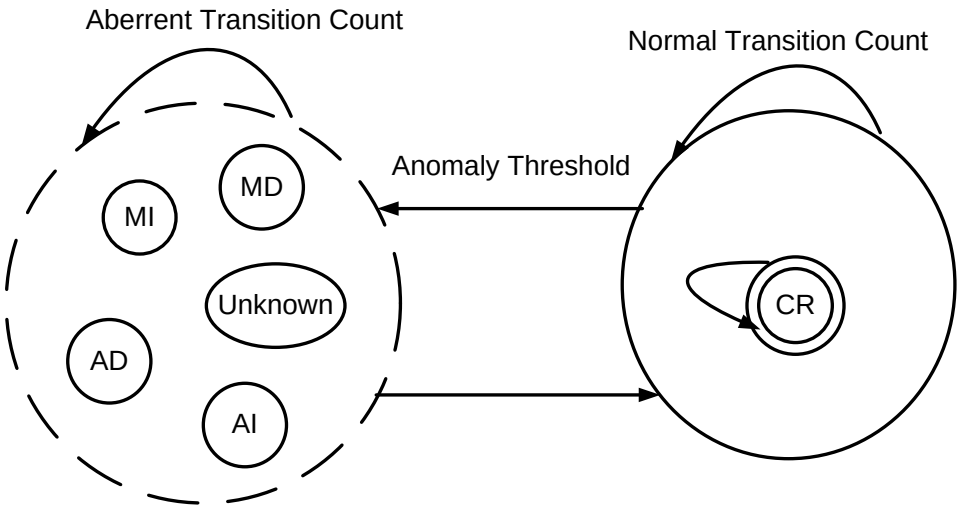


Figure 47: Simple TCP-Vegas FSM

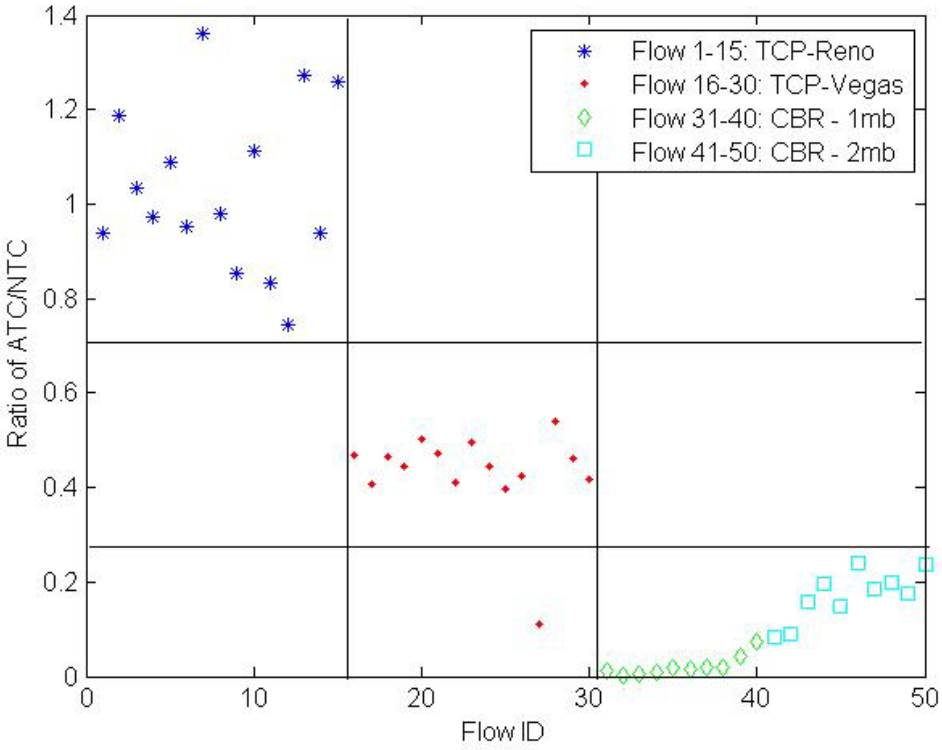


Figure 48: Detection outcome based on Vegas FSM

CHAPTER VII

CONCLUSION

Content and packet flow anomaly detections are important to cloud computing, which relies on networks to cope with the big data from different types of devices. To deal with content and packet flow anomalies, multiple approaches have been proposed, studied and verified in this dissertation.

For the detection of content-based anomalies, I proposed the PEC framework utilizing the competitive aging-scoring scheme to identify suspicious and bulk feature instances, which are later stored into the blacklist to detect the UNE. Considering the large amount of packets passing through the gateway, PEC takes O(1) time to update the score and age of onset feature instances given the limited memory. The birth of a feature instance starts from its first time arrival at the system, and the death of an instance occurs when its score is not updated and other instances come to occupy the whole age table. A probability model based on the Poisson process is built to derive PEC parameters consisting of the score threshold and age table size given the specified detection rate and latency. Multiple simulations have shown the accuracy of the model by collecting multiple samples of the detection rate and latency based on the model-derived S and M, where each sample is the mean of 100 runs.

To further verify the feasibility of PEC, the prototype of PEC is integrated with the sendmail application to detect URL links of UNE, albeit the system can be easily expanded to filter other features. URL links extracted from the corpus of spamming-

phishing messages are used to generate UNE messages, which are then mixed with regular messages to test the prototype. To meet the performance challenges, an O(1) spinning wheel algorithm is developed for the aging-scoring function. Experimental results on a Xeon based server show that PEC can handle 1.2M score updates, hashing and matching 7k URL links, and parsing of 200 messages bodies of average size 1.5kB. The lossy PEC detection system can be easily scaled up and down by progressive selection of detection features and detection thresholds. It can be used alone or as an early screening tool for existing infrastructure to defeat major UNE flooding.

For the packet flow anomaly detection, my goal is to detect the anomalous and undefined protocols using the inter-packet time as the input sample. The detection process consists of several important steps, which are RTT estimation, cwnd estimation, behavior state identification, and protocol detection. The method I proposed is based on the signal processing and different from model-driven methods, which depict the long-term behavior of flows from the perspective of statistics and probabilities. Due to quick changing of the congestion window, the dynamics of flow rates are the signal to determine the protocol. The model based on the average concept cannot capture and detect the transient change of the flow rate.

Because the flow rate is the congestion window size divided by the round trip, the estimation of RTT is the first step to infer the flow rate. By the study, we found that the traditional DFT method is incapable of estimating RTT within a reasonable error range. To reduce the estimation error, I proposed the EWMA Lomb periodogram approach, which considers the input noise reduction and unevenly spaced sampling

simultaneously. NS2 simulations have shown EWMA Lomb periodogram has more accurate RTT estimates than the DFT filter bank and genuine Lomb periodogram.

Recognizing that no estimation is perfect and errors can be accumulated from the RTT estimates to the cwnd estimates, several methods are studied in this dissertation. A low-pass filter is one of the widely used approaches to reduce the signal noises. If the estimation window is based on N RTTs, the variance of the estimation error can be decreased by 1/N. However, the low-pass filter is incapable of detecting the instant cwnd change such as MD, which typically occurs very shortly compared to AI. On the other hand, the high-pass filter cannot reduce the noise when the flow is in the AI state.

I proposed the approach using CUSUM banks to detect the behavior state of flows, where CUSUM is typically used to detect the change point of a signal sequence. CUSUM itself can filter the noise, and its algorithm parameters determine the detection sensitivity. By cascading a set of CUSUM banks, the relationship between different behavior states is created, where each CUSUM bank is in charge of one of the behavior states. Totally, eight CUSUM banks are created; four of them are for the cwnd increase detection and the residual four for cwnd decrease detection. A control policy is developed to reset different CUSUM banks and determine the current state.

Since the protocol behavior can be expressed by its behavior state change, e.g. TCP-Reno is designed to transit from MI to AI, but not AI to MI, the protocol-based FSM is created to detect if a transition is legal or not. By defining the anomaly threshold and the ratio R, the number of aberrant transitions over that of normal transitions, flows of undefined protocols can be identified in the R-based pattern. Multiple NS2

simulations based on the dumbbell topology have shown that (1) TCP-Reno flows and CBR flows can be classified using the R-based pattern, (2) TCP-Reno flows and TCP-Vegas flows can be differentiated, (3) TCP-Reno, TCP-Vegas, CBR1 and CBR2 can be differentiated using the Reno-based FSM, where the rate of CBR1 flows is different from that of CBR2, and (4) TCP-Reno, TCP-Vegas, CBR1 and CBR2 can be classified using the Vegas-based FSM, i.e. the proposed approach in this dissertation can be applied to different TCP flavors.

REFERENCES

[1] S. Y. Lin, C.C. Tan, J. C. Liu and M. Oehler, "High Speed Detection of Unsolicited Bulk Emails", in Proceedings of the ACM/IEEE Symposium on Architectures for Networking and Communications Systems, Orlando, FL, USA, December 3-4, 2007.

[2] "Snort Network Intrusion Detection System", http://www.snort.org.

[3] "Bro Intrusion Detection System", http://bro-ids.org./Overview.html.

[4] Defense Information Systems Agency, "DISA IAVA Process Handbook", Version 2.1. Washington, DC, USA, June 11 2002. http://www.tricare.osd.mil/contracting/healthcare/solicitations/TDP/0000/00_Attachment_16.pdf.

[5] "Application Layer Packet Classifier for Linux", http://l7-filter.sourceforge.net.

[6] F. Yu, Z. Chen, Y. Diao, T. V. Lakshman, and R.H. Katz, "Fast and Memory-Efficient Regular Expression Matching for Deep Packet Inspection", in Proceedings of the 2nd Symposium on Architectures for Networking and Communications Systems, San Jose, California, USA, December 3-5, 2006.

[7] S. Kumar, S. Dharmapurikar, F. Yu, P. Crowley, J. Turner. "Algorithms to Accelerate Multiple Regular Expressions for Deep Packet Inspection", in Proceedings of the ACM SIGCOMM Conference, Pisa, Italy, September 12-15, 2006.

[8] J. Moscola, J. W. Lockwood, R. P. Loui, M. Pachos, "Implementation of a Content-Scanning Module for an Internet Firewall", in Proceedings of the IEEE Symposium on Field Programmable Custom Computing Machines, Napa Valley, CA, April 8 - 11, 2003.

[9] B. C. Brodie, R. K. Cytron, and D. E. Taylor, "A Scalable Architecture for High-Throughput Regular Expression Pattern Matching", in Proceedings of International Symposium on Computer Architecture, Boston, MA, USA, June 17-21, 2006.

[10] C. Dwork, A. Goldberg, and Moni Naor, "On Memory-Bound Functions for Fighting Spam". in Proceedings of the 23rd Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17 - 21, 2003.

[11] "The Spamhaus Project", http://www.spamhaus.org/index.lasso

[12] "Know Your Enemy: Tracking Botnets", http://www.honeynet.org/

[13] "Domainkeys: Proving and Protecting Message Sender Identity", retrieved at Jan. 8, 2007, http://antispam.yahoo.com/domainkeys.

[14] "The Apache SpamAssassin Project", retrieved at Feb. 08, 2007, http://spamassassin.apache.org/

[15] "SpamAssassin Benchmark", retrieved at Feb. 14, 2007, http://www.isode.com/whitedissertations/spamassassinbenchmark.html

[16] I. Fette, N. Sadeh, A Tomasic, "Learning to Detect Phishing Emails", in Proceedings of the International World Wide Web Conference, Alberta, Canada, May 8-12, 2007.

[17] “Spamato Spam Filter System”, retrieved at Mar. 22, 2007, http://www.spamato.net/

[18] “Vipul’s Razor”, retrieved at Mar. 22, 2007, http://razor.sourceforge.net/

[19] “Earl Grey Filter”, retrieved at Mar. 22, 2007, http://www.spamato.net/

[20] “Domainator”, retrieved at Mar. 22, 2007, http://www.spamato.net/

[21] R. Segal, J. Crawford, J. Kephart, and B. Leiba, “SpamGuru:An Enterprise Anti-Spam Filtering System”, in Proceedings of the Conference on Email and Anti-Spam, Mountain View, CA, July 30-31, 2004.

[22] S. J. Stolfo, et al, “Behavior-based Modeling and Its Application to Email Analysis”, in Proceedings of ACM Transactions on Internet Technology, Volume 6, Issue 2, Pages 187-221, 2006.

[23] K. Wang, J. J. Parekh, S. J. Stolfo "Anagram: A Content Anomaly Detector Resistant to Mimicry Attacks", in Proceedings of International Symposium on Research in Attacks, Intrusions and Defenses, Hamburg, Germany, September 20-22, 2006.

[24] S. Venkataraman, et al, "New Streaming Algorithms for Superspreader Detection”, in Proceedings of the Network and Distributed System Security Symposium, San Diego, CA, February 3-4, 2005.

[25] F. Li, M. H. Hsieh, “An Empirical Study of Clustering Behavior of Spammers and Group-based Anti-Spam Strategies”, in Proceedings of the Conference on Email and Anti-Spam, July 27-28, 2006, Mountain View, CA, USA.

[26] D. Gao, M. Reiter and D. Song, "Behavioral Distance for Intrusion Detection", in Proceedings of the International Symposium on Recent Advances in Intrusion Detection, Seattle, WA, USA, September 7-9, 2005.

[27] S. Dharmapurikar, M. Attig, and J. Lockwood, “Deep Packet Inspection using Parallel Bloom Filters”, in Proceedings of the IEEE Symposium on High Performance Interconnects, Palo Alto, CA, USA, August 20-22, 2003.

[28] Y. Zhang, S. Singh, S. Sen, N. Duffield and C. Lund, “Online Identification of Hierarchical Heavy Hitters: Algorithms, Evaluations, and Applications", in Proceedings of the Internet Measurement Conference, Taormina, Sicily, Italy, October 25-27, 2004.

[29] G. Cormode et al., “Finding Hierarchical Heavy Hitters in Data Streams”, in Proceedings of the International Conference on Very Large Data Bases, Berlin, Germany, September 9, 2003.

[30] G. S. Manku, R. Motwami, “Approximate Frequency Counts over Data Stream”, in Proceedings of the International Conference on Very Large Data Bases, Hong Kong, China, August 20-23, 2002.

[31] B. Krishnamurthy, et al, “Sketch-based change detection: Methods, evaluation, and applications”, in Proceedings of the Internet Measurement Conference, Miami, FL, USA, October 27-29, 2003.

[32] R. Schweller, A. Gupta, E. Parsons, and Y. Chen, “Reversible Sketches for Efficient and Accurate Change Detection over Network Data Streams”, in

Proceedings of the Internet Measurement Conference, Taormina, Sicily, Italy, October 25-27, 2004.

[33] "SDBM", retrieved at Mar. 03, 2007, http://search.cpan.org/src/NWCLARK/perl-5.8.8/ext/SDBM_File/sdbm/README.

[34] Sheldon M. Ross, "Introduction to Probability Models", Academic Press, San Diego, USA. 2003.

[35] "Sendmail.org", retrieved at Mar. 17, 2007, http://www.sendmail.org/.

[36] "Berkeley DB", retrieved at Feb. 11, 2007, http://www.oracle.com/technology/products/berkeleydb/db/index.html.

[37] "2005 TREC Public Spam Corpus", retrieved at Jan. 16, 2007, http://plg.uwaterloo.ca/~gvcormac/treccorpus/about.html.

[38] P. Barford, J. Kline, D. Plonka, and A. Ron, "A Signal Analysis of Network Traffic Anomalies", in Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment, New York, NY, USA, November 2002.

[39] C. Estan, S. Savage, and G. Varghese, "Automatically Inferring Patterns of Resource Consumption in Network Traffic", in Proceedings of the ACM SIGCOMM Conference, Karlsruhe, Germany, August 25-29, 2003.

[40] T. Auld, A. W. Moore, and S. F. Gull, "Bayesian Neural Networks for Internet Traffic Classification", in Proceedings of IEEE Transactions on Neural Networks, Volume 18, Issue 1, Pages 223–239, January 2007.

[41] T. Karagiannis, K. Papagiannaki, and M. Faloutsos, "BLINC: Multi-level Traffic Classification in the Dark", in Proceedings of the ACM SIGCOMM Conference, Philadelphia, PA, USA, August 2005.

[42] C. Estan, K. Keys, D. Moore, and G. Varghese, "Building a better NetFlow", in Proceedings of the ACM SIGCOMM Conference, Portland, OR, USA, August 30 – September 3, 2004.

[43] A. Lakhina, M. Crovella, and C. Diot, "Characterization of Network-Wide Anomalies in Traffic Flows (Short Dissertation)", in Proceedings of the Internet Measurement Conference, Taormina, Sicily, Italy, October 25-27, 2004.

[44] A. Soule, K. Salamatian, and N. Taft. "Combining Filtering and Statistical Methods for Anomaly Detection", in Proceedings of the Internet Measurement Conference, Berkeley, CA, USA, October 19-21, 2005.

[45] T. J. Ott, T. V. Lakshman, and L. Wong, "SRED: Stabilized RED", in Proceedings of the Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'99), New York, NY, USA, March 21-25, 1999.

[46] A. Lakhina, M. Crovella, and C. Diot, "Diagnosing Network-Wide Traffic Anomalies", in Proceedings of the ACM SIGCOMM Conference, Portland, OR, USA, August 30 – September 3, 2004.

[47] M. Kodialam and T. V. Lakshman, "Detecting network intrusions via sampling: A game theoretic approach," in Proceedings of the 22nd Annual Joint Conference of

the IEEE Computer and Communications Societies (INFOCOM'03), Piscataway, NJ, USA, Apr. 1-3, 2003.

[48] J. Erman, M. Arlitt, and A. Mahanti, "Traffic Classification Using Clustering Algorithms", in Proceedings of the ACM SIGCOMM Minenet Workshop, Pisa, Italy, September 11-15, 2006.

[49] A. W. Moore and D. Zuev, "Internet Traffic Classification Using Bayesian Analysis Techniques", in Proceedings of the International Conference on Measurements and Modeling of Computer Systems, Banff, Alberta, Canada, June 6-10, 2005.

[50] A. Lakhina, M. Crovella, and C. Diot, "Mining Anomalies using Traffic Feature Distributions," in Proceedings of the ACM SIGCOMM Conference, Philadelphia, PA, USA, August 22-26, 2005.

[51] M. Roughan, S. Sen, O. Spatscheck, and N. Duffield, "Class-of-service Mapping for QoS: A Statistical Signature-based Approach to IP Traffic Classification", in Proceedings of the Internet Measurement Conference, Taormina, Sicily, Italy, October 25-27, 2004.

[52] H. Xie, Y. Yang, A. Krishnamurthy, Y. Liu, and A. Silberschatz, "P4P: Provider Portal for Applications," in Proceedings of ACM SIGCOMM Computer Communication Review, Volume 38, Issue 4, Pages 351-362, 2008.

[53] B. Rozovskii, A. Tartakovsky, R. Blazek and H. Kim ,"A Novel Approach to Detection of Intrusions in Computer Networks via Adaptive Sequential and Batch-Sequential Change-Point Detection Methods", in Proceedings of IEEE Transactions on Signal Processing, Volume 54, Issue 9, Pages 3372-3382, 2006.

[54] J. Padhye, V. Firoiu, D. Towsley and J. Kurose, “Modeling TCP Throughput: A Simple Model and Its Empirical Validation”, in Proceedings of ACM SIGCOMM Computer Communication Review, Volume 28, Issue 4, Pages 303-314, 1998.

[55] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery, “Numerical Recipes in C”, Cambridge University Press, New York, NY, 1992.

[56] Ryan Lance, Ian Frommer. “Round Trip Time Inference Via Passive Monitoring”, in Proceedings of ACM Sigmetrics Performance Evaluation Review - Special Issue on the First ACM SIGMETRICS Workshop on Large Scale Network Inference, Volume 33, Issue 3, Pages 32-38, ACM New York, NY, USA, December 2005.

[57] H. P. Van Dongen, et al, "A procedure of multiple period searching in unequally spaced time-series with the Lomb-Scargle method", Biological Rhythm Research, Volume 30, Issue 2, Pages 149-177, 1999.

[58] M. Basseville and I. V. Nikiforov, “Detection of Abrupt Changes: Theory and Application”, Prentice Hall, Upper Saddle River, NJ, USA, 1993.

[59] B. E. Brodsky and B. S. Darkhovsky, “Nonparametric Methods in Change-point Problems”, Kluwer Academic Publishers, Norwell, MA, USA, 1993.

[60] H. Wang, D. Zhang, and K. G. Shin. “Detecting SYN Flooding Attacks”, in Proceedings of the 21st International Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2002), New York, NY, USA, June 2002.