EVENT BASED MEASUREMENT AND ANALYSIS OF INTERNET NETWORK TRAFFIC

by

Sean Raymond McPherson

A Dissertation Presented to the
FACULTY OF THE USC GRADUATE SCHOOL
UNIVERSITY OF SOUTHERN CALIFORNIA
In Partial Fulfillment of the
Requirements for the Degree
DOCTOR OF PHILOSOPHY
(ELECTRICAL ENGINEERING)

August 2011

Dedication

To my parents, Ron and Lora McPherson, who are the foundation of the tremendous support structure built for me by my entire family. I know for certain that I would never have made it to the end of this daunting and rigorous path had it not be for the simple, yet imperative life lesson that you taught to me at an early age - if you commit yourself to something, you must not give up. I appreciate you for the simple actions, like our weekly phone calls that always brighten my spirits, to the feeling of strength I get just knowing that you support me and my goals no matter how lofty they may be. I appreciate you for always allowing me to be myself, and thank you for remaining at my side throughout this entire incredible journey.

To my sister and brother-in-law, Shalynn and Dale Taylor, whose small, but expanding family, has brought our entire family that much closer together. Being able to spend vacations with you and hearing stories about my growing nieces has provided me all the pleasurable experiences of raising a family of my own without the overwhelming commitment.

To my aunt and uncle, Tom and Carol McPherson, who always showed up at just the right time for our yearly adventure weekend. Providing a much needed break from everything along with some stimulating and, thankfully, non-research related discussion.

Acknowledgements

I would like to thank, first and foremost, my dissertation advisor Dr. Antonio Ortega for his helpful insights, educating advice and most importantly his tireless patience when dealing with me as a fledgling researcher.

I would also like to say thank you to all my colleagues and friends who I have had the pleasure to know during my studies. In particular I would like to express my gratitude to Gautam Thatte and Genevieve Bartlett for the many spirited discussions we shared, some of which even related to research.

Finally, I would like to thank Prof. John Heidemann, Prof. Urbashi Mitra, and Prof. Leana Golubchik for acting as my dissertation committee. The feedback I received in group meetings early on from Prof. Heidemann and Prof. Mitra, while at times difficult on my ego, significantly contributed to my process of becoming a researcher, and for that I am grateful.

Table of Contents

List Of Tables

List Of Figures

Abstract

Analyzing Internet traffic is critical to ensure the proper operation and maintenance of the current network infrastructure as well as to guide the expansion of future Internet pathways. Being able to analyze Internet traffic efficiently and with minimal error demands systems capable of measuring network traffic events while preserving characteristics of the traffic important to the particular analysis being conducted. Because of this it is important that measurement and analysis systems for Internet traffic be designed in a cooperative manner.

As a starting point of our cooperative measurement and analysis system design, we examine timing errors, with respect to a specific type of analysis task, inherent to Internet measurement systems. For select measurement systems we derive models for the timing errors, and show that with the proper choice of signal representation most of the timing errors can be mitigated. The signal representation we choose, called *SigVec*, is a modified point process representation. The modification is required because, in certain measurement systems, only a subset of packet arrival timestamps accurately reflect the packet timing. To preserve timing accuracy subsampling of the signal is required, which we incorporate in the *SigVec* representation. Then for specific Internet analysis tasks we propose a method to optimize the subsampling. Most importantly, our *SigVec* representation and subsampling optimization are very general and can be applied to *any* existing measurement system, including those like NetFlow that record flow measurements instead of packet arrivals, and measurement systems

designed to follow the IPFIX/PSAMP protocols recommended by the IETF [2,3,15].

Conveniently, our *SigVec* representation allows us to use the wealth of existing theory regarding point and renewal processes. In particular, from renewal theory we select a formulation, called the renewal density, which is suited for analyzing long range characteristics of Internet traffic. Using the renewal density formulation and our *SigVec* signal representation we derive a novel detection system, called inter-arrival based anomaly detection (IA^{2}D), for detecting low-rate periodic anomalies in Internet traffic. Because IA^{2}D uses the renewal density we can employ many features not found in existing systems. One feature of our system is the use of subdensities, which divide the renewal density into narrow time segments; this allows our system to detect and distinguish between multiple periodic anomalies, something most other detection systems are incapable of doing. Another feature of our system, due to renewal theory, is the ability to analyze and completely characterize the performance of our detection system for an idealized set of measurements, e.g., a Poisson process. Using this idealized analysis we derive expressions for system parameters, such as the time-to-detection, which we then use as guidelines for selecting detection system parameters when actual Internet network traffic is analyzed. Our system differs from state of the art systems by: i) detecting periodic anomalies at lower-rates, ii) detecting anomalies in aggregate Internet traffic, i.e., without flow separation as used in some systems, and iii) being able to distinguish between multiple periodic anomalies.

Chapter 1:

Introduction

Internet traffic measurement systems are used in numerous network analysis applications such as network tomography, anomaly detection and measuring quality-of-service [54, 61]. In general a measurement system is any system connected to an Internet link that extracts information from packet headers, such as packet size, as the packets pass through the system. The measurement system records this information along with a timestamp indicating, with some processing delay, when the packet arrived at the measurement system. In addition to measurement systems that simply record packet header and timestamp data, Internet traffic measurement systems can use information extracted from the packet header to construct flow based measurements. A flow is a connection between a pair of computer systems. Packets from the same flow are identified by having the same source and destination IP addresses, as well as using the same port connections [15].

Both packet and flow level measurements are useful for Internet network traffic analysis in a variety of applications. Internet traffic measurements are used to analyze existing networks to identify under-performing and outdated links, which can cause bottlenecks in otherwise healthy networks and should be repaired or updated for optimal network performance [9,25,42,47]. Detecting anomalies is another important task that makes use of network traffic measurements to detect denial-of-service (DOS) attacks, which can affect availability of web sites and services thereby impacting the economy of the Internet [12,34,36,74]. In addition to being important for maintenance and security of the Internet, network measurements can also effect its operation. For instance, network measurements are used in the congestion avoidance algorithms of some recent variations of the transmission control protocol (TCP), such as TCP Vegas and FAST TCP. The congestion avoidance algorithm of TCP Vegas uses the average round trip time (RTT) between when a packet is sent and when it is acknowledged as being received [13]. In FAST TCP queuing delay replaces the RTT in the congestion avoidance algorithm [8]. Both RTT and queuing delay are estimated using timing information obtained from packets, therefore, accurate measurements are important to the proper operation of these congestion avoidance algorithms.

Clearly, being able to measure Internet packet timing with minimal error is important to Internet traffic analysis and operation. The focus of the work in this thesis is on generating Internet traffic measurements while mitigating timing errors, yet at the same time preserving characteristics of the traffic important to a specific type of analysis task. We propose that we can accomplish these two criteria by selecting the proper signal representation, and through the use of subsampling. The signal representation is the form in which the measurement system records the Internet network traffic, in this dissertation we select a signal representation that is most natural to Internet network traffic events, namely a point process representation. Subsampling is used when the timing information of particular measurements is known to be inaccurate, typically due to the measurement system. By modifying the operation of the measurement system it is possible to optimize which measurements are inaccurate, and thus which measurements are subsampled. By optimizing the measurement system we create measurements that preserve the desired traffic characteristics for particular analysis tasks.

To illustrate how the signal representation and subsampling optimization can benefit Internet traffic analysis we consider a typical, and important, analysis application, detecting anomalies in aggregate Internet traffic. In this dissertation we select a particular anomaly, low-rate periodic traffic, and design a detection system that directly uses our signal representation instead of modifying the representation as done in similar detection systems [12, 34, 74]. Further, we optimize subsampling of the traffic measurements specifically for the low-rate anomaly detection task, which allows us to process large collections of Internet traffic efficiently while maintaining reliable detection performance. We test the performance of our detection system in multiple applications against that of other detection systems that require modifying the signal representation [74] or separating the traffic based on flows [10]. The performance results indicate our system performs comparably to other systems, however, since we require no modifications to the aggregate signal our system can be used more readily. For example, unlike the method in [10] our method works when flow information is not available because an attacker uses IP address spoofing.

1.1 Motivation

The motivating theme for the work in this dissertation is the importance of *generating Internet measurements with minimal timing error while preserving characteristics of the traffic important to specific analysis applications.* Typical Internet traffic events, e.g., packet arrivals or the beginning of a flow, occur discretely, and measurements of these events are recorded by standard systems with individual timestamps for each event recorded. However, many Internet traffic analysis techniques convert the event based measurements to a more common time series representation in order to be compatible with standard signal processing techniques (e.g., fast Fourier transform, power spectral density, etc.). The conversion from the event based measurements to a time series signal representation is a potential source of timing error, and requires additional computations. By selecting one signal representation compatible with both the measurement and analysis systems we are able to mitigate a potential source of timing error.

We derived our signal representation, named *SigVec*, based on our analysis of measurement systems, particularly software based systems, which is described in Chapter 2. Software based measurement systems are a desirable alternative to hardware system because they are low-cost. However, certain timing errors inherent to these systems make them unsuitable for Internet traffic analysis, particularly when used on high speed networks. One source of error, called interrupt coalescence, groups packets together creating a situation where out of the entire group of packets the timestamp is accurate for only one packet. In a sense interrupt coalescence can be seen as a form of subsampling, and in our signal representation only one accurate timestamp for the group of packets is recorded and the other timestamps are subsampled. By optimizing how interrupt coalescence works, as discussed in Section 2.3, we are able to generate subsampled measurements that preserve desired characteristics of the Internet traffic making the measurements more useful for Internet traffic analysis. This optimization is task specific, and using our optimization method the measurements generated are signal independent and do not contain the timing bias found in standard interrupt coalescence methods.

It is important to note that, as discussed in Section 2.4, the *SigVec* signal representation and subsampling optimization techniques are general and can be used in any existing measurement system. In particular, subsampling is important for all measurement systems to reduce the amount of data that must be stored, and to reduce the computational cost in analyzing Internet traffic.

In Chapter 4, we design techniques for Internet network traffic analysis that use our *SigVec* representation in order to reduce the timing errors associated with transforming the signal to a time series representation. Many analysis applications, anomaly detection for instance, work by analyzing the long-term timing behavior in Internet traffic. We use concepts from renewal theory, specifically the renewal density, to perform such long-term timing behavior analysis using our signal representation. As an example application of the possible uses of long-term timing analysis we selected low-rate periodic anomaly detection and designed a detection system, which we called inter-arrival based anomaly detection or IA^{2}D, as described in Section 3.3. This detection system is shown in Sections 4.3 and 4.4 to perform as well as similar detection systems, with some additional benefits including: (i) ability to distinguish multiple periodic anomalies and (ii) being amenable to subsampled measurements.

Finally, in Chapter 5 we present a multi-level, distributed forensic analysis system. The multi-level system varies the level of subsampling, with a high subsampling rate used at the first level to quickly determine measurements likely to contain an anomaly, and then lower subsampling rates at the next level to improve detection confidence. The multi-level architecture of this system requires that the measurement system and analysis application work together, which occurs seamlessly with our *SigVec* signal representation and IA^{2}D detection system.

In this dissertation we show that by selecting the proper signal representation and by designing Internet traffic analysis techniques that use this representation, we can mitigate many timing errors and integrate the measurement and analysis systems. Further, by optimizing subsampling in the measurement system we can preserve characteristics of the traffic important for various Internet traffic analysis applications, while reducing the data storage and computational costs required to perform such analysis.

Given the generality of our signal representation and subsampling optimization method there exist many applications beyond what is considered in this thesis for further research, both within Internet traffic analysis and in other areas. For example, the techniques derived in this thesis for Internet traffic can be applied to signals from research areas, such as biological or financial data analysis, where event based signals occur naturally. Two examples of event based signals from these areas are the firing of neurons in the human body, and the purchase and sale times of stocks in the stock market [60]. Further, low-rate periodic events are just one type of anomaly currently found in Internet network traffic. Low-rate periodic events were considered here because, by using the renewal density, it is straightforward to distinguish between Internet traffic that does and does not contain periodic traffic. It is possible to detect other types of anomalies by modifying our IA^{2}D detection system, however, depending on the characteristics of the anomaly, distinguishing between the renewal density of Internet traffic with and without the anomaly may not be as straightforward as in the periodic case.

1.2 Summary of Contributions

The following list provides a brief summary of the key contributions of this dissertation.

- Software Measurement System Modeling, Section 2.2 - Similar measurement system analysis has been performed previously, for example the work of Salah that derived the maximum throughput and average timing delay of packets through a software measurement system [66]. However, the analysis presented here is different in that it provides system component models which reflect the actual operation of the measurement system components, yet are simplified, making system analysis easier. Using this analysis we show that with the choice of our signal representation we can mitigate many of the timing errors inherent to the software measurement system.

- *SigVec* Signal Representation, Section 2.2.4 - The signal representation derived here contains all accurate information captured by the software measurement system. As mentioned previously, due to interrupt coalescence, only one timestamp per group of packets is accurate. Therefore our signal representation reflects this and provides one measurement per group of packets; each measurement indicating the one accurate packet timestamp and the number of packets contained in the measurement. The signal representation with two components is a novel idea that conveys more information about the characteristics of the subsampled Internet traffic than is indicated by just the subsampled timestamps alone.

- Optimized interrupt coalescence, Section 2.3 - The operation of interrupt coalescence is, for the first time, optimized specifically for Internet traffic analysis. In order to optimize the operation of interrupt coalescence we define a metric, which can be modified for specific analysis applications, and use this metric to optimize the probability distribution of the number of packets per measurement found in the interrupt coalescence algorithm used. Compared to existing interrupt coalescence methods our optimized interrupt coalescence technique is signal independent and does not incur timing bias found in some existing methods. The optimization metric and technique are completely general and can be used to design subsampling in other measurement systems, not just those that use interrupt coalescence.

- IA^{2}D Detection System, Chapter 3 - The detection system designed here uses a renewal theory formulation called the renewal density, which allows us to analyze long-range timing behavior using our signal representation. Because the system works in the time domain we can detect and *distinguish* between multiple periodic anomalies, which is not possible in many other detection methods. Distinguishing multiple periodic anomalies is possible because we independently analyze different time segments of the renewal density, which we call sub-densities.

- T_D - Time-to-Detection Analysis, Sections 4.3.1 and 4.4.1 - IA^{2}D is designed to wait a specific amount of time in order to gather enough evidence to determine if an anomaly is present or not. The length of time to gather evidence is called the time-to-detection, T_D, and using renewal theory we derive expressions for T_D for two different anomaly detection applications. The expressions for T_D are based on the assumption that the background Internet traffic is modeled by a Poisson processes, therefore, we make certain modifications to the expressions for real Internet traffic in Chapter 4, where we also show that the modifications reliably predict the T_D required for real Internet traffic. The T_D analysis also provides an addition benefit in the implementation of our IA^{2}D detection system. For each subdensity in our system T_D is different, however, for a better user interface we desire to have all subdensities report detection decisions at the same time. Therefore, we design a jumping window implementation, based on the T_D calculation, such that each of the subdensities can report detection decisions at the same time, at a rate determined by the user.

- Multi-Level Detection System, Chapter 5 - We designed as a final application of IA^{2}D a distributed, forensic anomaly detection system, which combines our work on measurement and analysis systems into one integrated application. In this architecture, multiple nodes record measurements of Internet traffic. When an anomaly is detected somewhere in the network, the recording nodes send their measurements to a processing node for forensic anomaly detection, seeking to determine the path of the anomaly through the network. In order to reduce the cost of transmitting data between recording and processing nodes we propose a multi-level detection method, which at the first level finds measurements that likely contain an anomaly, then request additional data to confirm the detection. The multi-level structure also speeds analysis at the processing node making such a system desirable for implementation in large, distributed measurement networks.

1.3 Outline

This dissertation is organized in the following manner. In Chapter 2 we examine software based measurement systems, construct a suitable yet simple timing error model for measurement generation in such a system and derive a signal representation and subsampling technique that mitigates the timing error. Using this signal representation, combined with ideas from renewal theory, we derive a low-rate periodic detection system, called IA^{2}D, in Chapter 3. Applications of IA^{2}D are studied in Chapter 4, and for each application a thorough system analysis is conducted using an ideal renewal process, e.g., a Poisson process. Finally, in Chapter 6 we provide some concluding remarks and highlight a number of potential applications of IA^{2}D in a broad set of research areas.

Chapter 2:

Signal Representation and Subsampling Optimization in Internet Measurement Systems

2.1 Introduction

As described in Chapter 1 measuring Internet traffic in real time is a task with numerous applications, many of which are important to the operation and maintenance of the current network infrastructure as well as to the expansion of future Internet pathways. The accuracy of the timing information affects the usefulness of the Internet measurements for many analysis tasks, including those mentioned in Chapter 1. The timing information may be used explicitly, as in the packet-pair method for estimating bottleneck bandwidths, which measures the inter-arrival time between pairs of probe packets [47]. Otherwise the timing information might be used implicitly, such as in DoS detection mechanisms that compute the autocorrelation and power spectral density of the incoming packet stream, searching for periodic components which could indicate an anomaly [34, 36]. Distortions or delays in the timing information recorded in Internet traffic measurements impact the performance of many analysis applications.

Clearly, measuring Internet traffic in real time is fundamentally important to Internet analysis, therefore, much attention has been given to designing methods and systems to generate measurements. Not all measurement systems generate timestamps for the recorded measurements, for example some systems count the number of packets per second matching some selection criteria (packet size, destination port number, etc.) and the measurements are packet counts output at regular intervals. The focus of this work however, is on systems that produce a timestamp that indicates the timing of the corresponding measurement. The notion of timestamping a measurement is important because it distinguishes between classes of measurement systems. Measurement systems can be categorized by the rate at which they are capable of producing measurements with accurate timing information. Cost is another way to classify measurement systems, and is linked to the accuracy of the timing information the system produces. The most accurate systems can cost as much as \$10,000, while less accurate systems, like those built using everyday personal computers, may cost only a few hundred dollars. When determining the required accuracy of the measurement system and the associated cost, it is important to consider the intended Internet analysis application. For many analysis applications lower cost systems, with less accurate timing information, or subsampling the number of output measurements, may be suitable as long as the effect of the measurement is known.

One class of measurement system, which we broadly define as hardware based systems, uses a very accurate hardware clock to generate timestamps for measurements. The use of a hardware clock is beneficial because it allows the timestamp to be generated before any processing by the host system takes place. By generating a timestamp before host system processing occurs, and thereby avoiding delays caused by the system processing, a hardware based system is capable of generating timestamps for individual packets at very high rates. Also, because the host system processing delays are removed, the timestamp indicates the time when the packet arrives at the measurement system. The accuracy of hardware based systems, most notably the DAG network cards developed by Endace [23, 24], depends on the precision of the hardware clock. Current systems feature hardware clocks capable of generating timestamps with nanosecond precision, which is sufficient for current high speed networks, e.g., 10 Gbps. The major drawback of hardware based systems is the price; where current systems cost on the order of $10,000. Further, as the speed of Internet network links increase, hardware clocks capable of generating timestamps with greater than nanosecond precision will be desirable; thereby driving up the cost to produce hardware based measurement systems. Note that it is possible to create lower cost hardware measurement systems that use less precise hardware clocks, or use subsampling to reduce the number of measurements, thus decreasing system complexity. The advantage of such lower cost hardware systems would be that they remove the timing delays due to system processing, as discussed in the next paragraph, however, currently such systems do not exist.

Due to the high cost of current hardware systems, recently there has been more research done on the viability of using alternative classes of measurement systems to conduct Internet traffic analysis. An alternative class of measurement system, which we define as software based systems, uses the host system clock to generate measurement timestamps. Costing less and being readily available, software based measurement systems represent a potential alternative to hardware systems, however, their fundamental drawback is the rate at which software systems can generate measurements. In a software based system the timestamp is not generated until the measurement software running on the host system receives the packet, which does not happen until after the packet has been processed by the network interface card (NIC). Processing in the NIC creates multiple, non-constant delays that impact the accuracy of the timing in software based measurements. The timing delays occur because, except during low-rate traffic conditions, a software system cannot generate individual timestamps for each incoming packet. Instead packets get queued, waiting to be processed, or more commonly a technique known as interrupt coalescence is used to reduce the burden on the system processor that is created by the network interface card requesting each packet to be processed individually. Interrupt coalescence reduces the burden by grouping multiple packets for the system to process together; reducing the time spent by the system context switching and other tasks. These are done once for each group of packets, but would be done for each packet if no interrupt coalescence was used [66]. Because multiple packets are grouped together, the timestamps recorded by the system indicate the time when processing by the NIC has been finished for the entire group of packets and not the arrival time of the individual packets, as in the hardware system case. Only when the incoming traffic rate is low enough can a software system produce timestamps for individual events; thus, the differentiation between hardware and software systems is based on the rate at which they can generate measurements.

The goal of this chapter is not to redesign software measurement systems in order to keep up with the increasing network rates. Instead we focus on two methods to mitigate the timing errors inherent to software measurement systems such that existing systems can be used for Internet network traffic analysis. The two methods we use to mitigate the timing errors are subsampling of measurements, and the choice of how to represent the measured signal. To derive a meaningful signal representation and determine methods to subsample measurements we first perform a thorough analysis of the effect of the software measurement system, and its inherent delays, in Sections 2.2.1, 2.2.2 and 2.2.3. Then by understanding these delays a new signal representation is derived in Section 2.2.4, which is able to group multiple packets (those received during the same interrupt) into a single measurement. It will be shown that the timing for one packet in each group is accurate; thus the new signal representation contains all the reliable measured information, one accurate timestamp and a count of the number of packets in the measurement. Given this system analysis and signal representation, we show, in Section 2.3, how to optimize the operation of interrupt coalescence for specific network traffic analysis tasks. Such an optimization is, we believe, the first of its kind, as normally the operation of interrupt coalescence is designed only to maintain an average interrupt rate. Finally, we will show in Section 2.4 that our signal representation can also be applied to measurements from high rate (e.g., hardware based) measurement systems, and that our method for optimizing interrupt coalescence can be used to optimize subsampling of measurements in general systems.

2.1.1 Related Work

Several approaches have been proposed to mitigate the effects of the software system processing. One method to reduce the timing delays associated with software systems is to improve the processing performance of such systems. The work of Varenni, *et al.* [76] and Dashtbozorgi [19] seeks to use multiprocessor software based measurement systems that can process multiple packets in parallel. Parallel packet processing removes or reduces the time that packets are queued waiting to be processed, and makes interrupt coalescence unnecessary. While such methods are possible given the current Internet link speeds, in the future such multiprocessor systems will become increasingly more complex to optimize in order to maintain pace with the bandwidth of the Internet. Additionally, the work in this thesis uses standard measurement systems because they cost less and are more available than the specially designed multiprocessor systems.

Another method to make software systems viable for Internet traffic analysis is to adapt network traffic analysis methods to software based measurements. Some recent work has focused on adapting a specific analysis task, bandwidth estimation, to work with software system measurements. Prasad, *et al.* [61] studied the effect of the measurement system on a standard bandwidth estimation technique, the packet train method, and based on their observations were able to adapt the estimation technique to mitigate the measurement system effect. Man, *et al.* [50] went a step further and adapted the analysis method by altering how the packet train was injected into the network. By altering the packet train the effect of the measurement system was reduced, which improved estimation time and performance. Thus most existing work that uses software based measurement systems involves modifying the analysis task to counteract the effect of the measurement system.

The work presented in this thesis begins with a similar idea, but with one key distinction. The previous work focused on how the measurement system impacts the signal used for their particular analysis task, i.e., bandwidth estimation. Our work instead looks at the entire measurement process, including how measurement timestamps and the signal used for analysis are generated. By studying the impact of the software system on the entire measurement process, we developed a completely different signal representation. Finally, using this new signal representation, we can perform various Internet analysis tasks with measurements generated using measurement systems, such as the software based system, that can only produce measurements at lower rates.

2.2 Software Measurement System Modeling

High rate measurement systems, e.g., hardware systems, are able to produce accurate timestamps for individual packets, even when the packets are received at a very high rate. The advantage of a hardware based system as compared to a low rate, or software based system is illustrated in Figure 2.1. The measurement timestamp generated by the hardware system occurs before the packet is processed by the NIC. This is advantageous because processing that takes place in the NIC creates multiple, variable delays that impact accuracy of the measurement timestamp. While the benefit of the hardware based system is apparent the drawback is in the price of the systems. Further as the speed of Internet networks increase the cost to produce hardware systems capable of matching the speed of the network will increase as well.

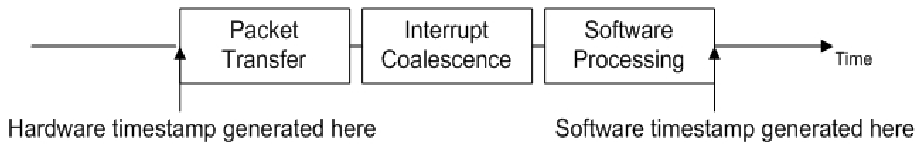
Packet Transfer
Interrupt Coalescence
Software Processing
Time
Hardware timestamp generated here
Software timestamp generated here

Figure 2.1: Packet Timestamping in Hardware and Software Measurement Systems

Software based measurement systems are one example of measurement systems that are not always capable of producing measurements for individual packet arrivals. The capability of software systems to produce measurements for individual packet arrivals depends on the incoming traffic rate and the maximum interrupt rate that they system can handle. However, software systems are a readily available, lower-cost alternative to hardware systems. The accuracy of the timestamps generated for a software based system depends on the processing done on the packet before a timestamp is recorded. As shown in Figure 2.1, the additional processing can be grouped into three separate blocks. The first block, described in Section 2.2.1, is a packet-size dependent delay that occurs when transferring the packet data to memory. The second block is interrupt coalescence (IC), which depends on the arrival timing of the incoming packet stream. Interrupt coalescence, covered in Section 2.2.2, is a method used in high speed network interface cards (NIC), which lessens the processing burden on the host system by reducing the number of interrupt service requests generated by the NIC indicating to the system that a packet is ready for processing. The last block is the delay due to software processing, which we discuss in Section 2.2.3. The software processing delay is a combination of multiple delays from the operating system as well as the application generating the timestamp.

Note that hybrid hardware/software measurements systems exist; one example system is CoralReef developed by CAIDA [43], which is actually a software suite designed to work with a number of high end network adapters. Using CoralReef along with a high end network adapter and host system measurements can be generated for individual packets without incurring the signal-dependent timing delay due to interrupt coalescence. Thus with such a system the only timing delays are packet size dependent, and can therefore be removed by post processing making such measurements reliable for Internet analysis. Because of the cost of components, high-end, hybrid measurement systems typically cost as much as commercial hardware systems and generally are designed specifically for and maintained by research laboratories or universities (CAIDA is based at the University of California's San Diego supercomputing facility). In our work we do not consider hardware or hybrid systems because of their associated cost.

2.2.1 Packet Transfer to Memory

In a software based system the incoming packet data is first transferred to system memory. The NIC waits for this transfer to complete before asserting a system interrupt letting the CPU know that the packet data is available for processing. Commonly, two methods are used to transfer data from the NIC to memory, direct memory access (DMA) and programmed input output (PIO). With PIO the CPU is used to transfer the data from the NIC to memory. This often overburdens the CPU, which is already stressed in a high speed network measurement system, so in this chapter discussion is limited to the study of DMA data transfers [63].

The delay, $D[n]$, incurred in transferring the packet data to memory is a combination of two delays, i.e., $D[n] = D_1[n] + D_2[n]$. The first delay, $D_1[n]$, is the time spent waiting to access the peripheral component interconnect (PCI) bus (used to transfer data). In typical systems the transfer rate of the DMA channel is comparable to that of the network link being measured, therefore, in our work we assume that $D_1[n]$ is negligible. As a particular example we consider the rates of the current DMA engines and current network link rates. According to [65] "a typical DMA engine can sustain over 1 Gbps of throughput across the [PCI] bus", which is sufficient for a Gigabit Ethernet NIC. For higher speed NIC, such as 10 Gigabit Ethernet, a PCI express bus can be used offering greater throughput than the PCI bus. Maximum raw throughput for the PCI express bus is 2.5 Gbps with actual data rates depending on the payload size (packet size) and ranging from 1.5 to 2.0 Gbps [38]. These values are for individual PCI express bus channels or lanes, therefore 10 Gigabit Ethernet NIC cards require multiple PCI express bus channels [69].

The second delay, $D_2[n]$, is the length of time required to actually transfer the data to memory. Because the speed of the DMA channel is fixed, the amount of time required to transfer a packet depends on the size of the packet itself. Thus, the delay, $D_2[n]$, is some constant multiple of the packet size. We approximate delay using the equation $D_2[n] = PacketSize/BitRate$ seconds. Here $BitRate$ is the bit rate of the DMA channel and $PacketSize$ is the size in bits of the current packet. This approximation for $D_2[n]$ captures the packet size dependence of the DMA transfer time.

Figure 2.2 shows an example of the packet size dependent data transfer model along with the our approximation of the model. In our approximation the delay $D[n] = D_1[n] + D_2[n]$ is simplified by the expression $D[n] = D_2[n]$. We use this approximation because, as discussed previously $D_1[n]$ is assumed negligible. One might argue that this approximation is overly simplistic, and that the DMA transfer time depends on the availability of system resources like the PCI bus. However, according to Salah, *et al.* in [66] "the transfer rate of incoming traffic [...] is not limited by the throughput of the DMA channel." Therefore, using $D[n] = PacketSize/BitRate$ our packet transfer model approximation accurately reflects the packet size dependence of the DMA transfer delay while being simple to implement in simulation.

In Figure 2.2 $m[n]$ indicates the time stamp of the beginning of a packet received at the NIC. At the output of the packet transfer block (in Figure 2.1 the timestamp of the incoming packet is now $m'[n] = m[n] + D[n]$). Using our approximation for $D_2[n]$ (lower section of Figure 2.2) the timestamp of the packet now indicates the end of the packet, and is given by $m'[n] = m[n] + PacketSize/BitRate$.

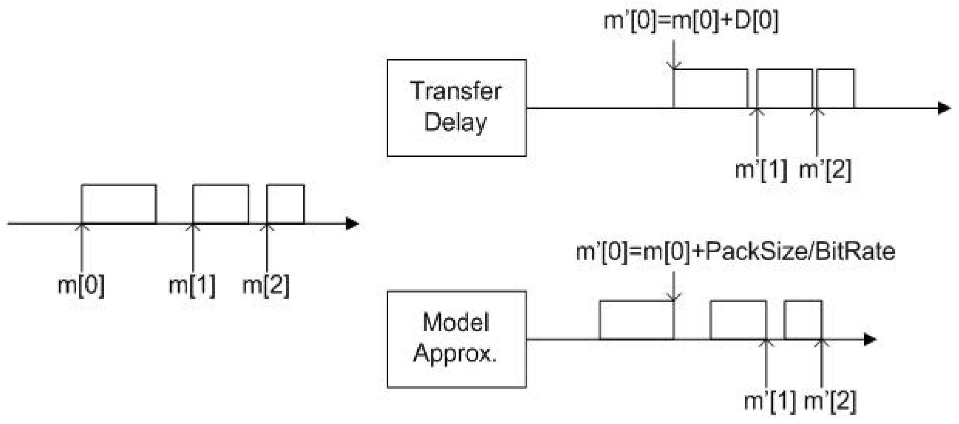

Figure 2.2: Example of Packet Size Dependent Transfer Model vs. Approximation of Model

2.2.2 Interrupt Coalescence

In a software based measurement system, after the packet data has been transferred to memory the NIC asserts a system interrupt informing the CPU that the packet is ready for processing. In low speed networks, an interrupt is asserted for each packet that the NIC receives, since at these speeds, e.g., 10 Mbps to 100 Mbps, typical CPUs are capable of responding to an interrupt service request (ISR) for each packet. However, in high speed networks, e.g., 1 Gbps and beyond, the number of packets received increases significantly and consequently the number of ISRs made by the NIC increases as well. The increase in ISRs can result in a condition called receive livelock where the host system cannot perform any processing and spends all of its clock cycles servicing the interrupt requests. To prevent livelock, a technique called interrupt coalescence (IC) is used in high speed NICs that reduces the workload of the host CPU by lowering the number of interrupts issued by the NIC [66].

Three variations of interrupt coalescence are commonly used: timer-based IC (TIC), packet-based IC (PIC), and hybrid time based methods (HIC), such as those found in the Intel NIC [37, 44, 66]. The goal of interrupt coalescence is to reduce the interrupt rate imposed on the measurement system by the NIC. The intuitive way to reduce the interrupt rate is to not issue an interrupt for each packet received. When an interrupt is issued depends on the method of interrupt coalescence. For example, when using PIC after the initial packet arrives, the NIC card waits until a given number of subsequent packets are received before an interrupt is asserted [66]. The limitation of this method is that during low traffic rate conditions the latency between the first packet arrival and interrupt assertion becomes excessive. Alternatively, TIC waits a fixed amount of time after the arrival of a packet before issuing an interrupt. In low traffic rate situations long fixed time values can also lead to excessive latency for TIC. Figure 2.3 shows how TIC and PIC generate interrupt service requests. On the left TIC is used with parameter T_{fix}, which is the length of time after the initial packet arrival before an interrupt is issued. Similarly on the right, PIC is pictured where the number of packets received before asserting an interrupt, C_{fix}, is set to 5.

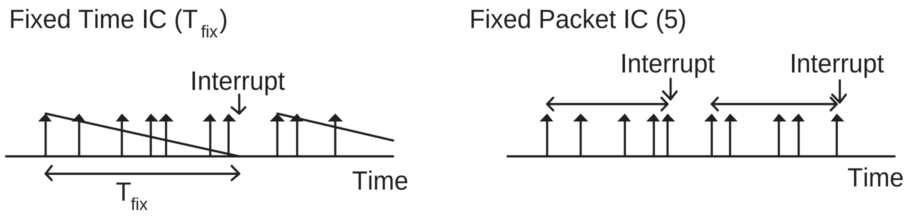

Figure 2.3: Example of interrupt coalescence using TIC and PIC methods

Although these simple methods successfully reduce processor burden, their limitations create a need for hybrid methods such as that developed by Intel, which incorporates two separate mechanisms to control the number of interrupts asserted on the system. These mechanisms are controlled by two separate timers named the absolute and packet timers [37]. Figure 2.4 illustrates the difference between the timer variations. The packet timer, T_{pack}, begins counting down after a packet has been received. If additional packets are received before the timer finishes counting down, then T_{pack} is reset and begins counting down again. Anytime T_{pack} reaches zero before a packet arrives to reset the timer, the NIC will assert an interrupt and all packets that were received since the previous interrupt will be processed by the CPU. During low traffic rate conditions the packet timer prevents excessive latency in the packet processing.

However, under high traffic rate situations when packets arrive in rapid succession it is possible that the packet timer will never count down to zero, so a second timer called the absolute timer, T_{abs}, is used. Similar to fixed time IC, the absolute timer waits a fixed interval following the receipt of an incoming packet before asserting an

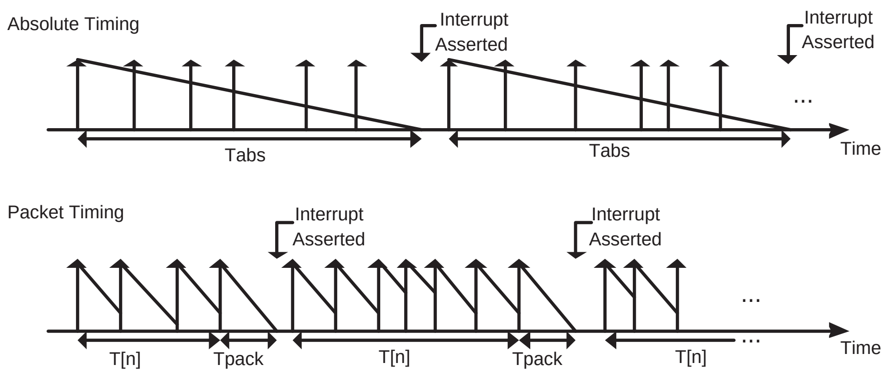

Figure 2.4: Examples of interrupt coalescence with absolute and packet timers controlling the assertion of interrupts

ISR. All packets received during the interval T_{abs} are processed by the system during the same ISR as the initial packet. Interrupt service requests are issued whenever T_{pack} or T_{abs} count down to zero; in this way interrupts are asserted efficiently during both high and low traffic rate conditions.

While the effect of the data transfer delay was packet size dependent, the effect due to interrupt coalescence is packet timing dependent. Because the incoming packet timing is not known in general, studying the effect of IC on individual packets is difficult. In [64] Salah used a Markov model to study the average timing delay experienced by individual packets in a system using fixed-packet interrupt coalescence. In our work, rather than study the effect of interrupt coalescence on individual packets, we propose a simplification in Section 2.2.3 and our signal representation in Section 2.2.4, which allows us to analyze the effect of IC on a group of packets (one measurement) instead of individual packets. The simplification and change of representation not only facilities the analysis, but also removes some timing errors, that are described in the next section, created by the measurement system. While our signal representation produces a subsampling in the number of data points it preserves only the data for which accurate timing information is known.

2.2.3 System Processing

After interrupt coalescence generates an interrupt service request for a group of packets, each packet must be processed by the system. The packet processing consists of two parts. The first part of processing is handling the ISR, which for systems using DMA consists of interrupt-context switching and informing the system kernel to begin protocol processing on the packet. Similar to the notation in [65], we define r to be the ISR handling rate, so that $1/r$ is the average time to handle an ISR. One of the benefits of interrupt coalescence is that the ISR processing is done for the entire group of packets, so the ISR servicing time for all the packets is the same as the ISR service time of a single packet [65]. The second part of processing involves protocol processing, based on the type of packet received (TCP, UDP, etc.), and user application processing, i.e., generating the time stamp. Unlike the first stage, the second stage of processing is done individually for each packet even if they are received as a group due to IC. In general, the processing done in the second stage depends on the availability of the system processor, which may be different for each packet. Therefore the system processing time is variable and we define it for the n-th packet as $S[n]$. Note that system kernels exist which can generate timestamps for the user application avoiding some timing errors, however, as will be discussed later, using our signal representation the effect of user application processing is removed in the final measurement timing.

One cause for variable system processing time is the fact that the second stage of processing may be interrupted at any time for ISR servicing. In systems without interrupt coalescence this fact makes it difficult to model the variability of the system processing delay. As shown in Figure 2.5, while the actual packet servicing time might be the same for all packets, the effective packet servicing time can be quite different from packet to packet and depends on the number of ISRs handled during packet processing.

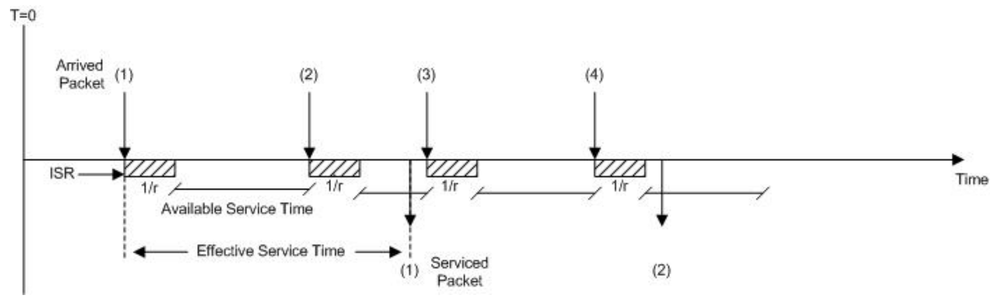

Figure 2.5: Example showing system processing operation in a system without interrupt coalescence

Alternatively, in measurement systems that employ interrupt coalescence the processing is slightly different, as shown in Figure 2.6. In this scenario the ISR is not issued until after the last packet in the coalesced group has been transferred to memory (here 3 packets are grouped together via IC). Then a single interrupt service request is handled for the 3 packets (taking $1/r$ seconds), and then packet processing begins. We assume that the system is able to process all the packets from the current coalesced group before an ISR is generated signaling that the next group of packets is ready for processing. Because of this assumption the second stage processing is assumed to be constant for each packet, so $S[n] = S$. When this assumption does not hold, the timestamp values in the final output measurement will suffer from jitter.

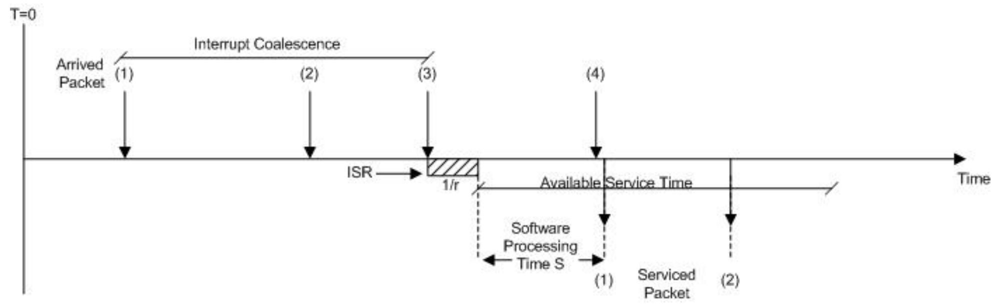

Figure 2.6: Example of system processing operation in a system with interrupt coalescence

Given this system processing delay model, and assuming the second stage processing is a fixed time S, the time difference between packets received in a single coalesced group would be exactly S. An example of this is shown in the top of Figure 2.7, where the packet timestamps are such that $M[0] - M[1] = M[1] - M[2] = S$. This fixed inter-arrival timing is artificial and does not reflect any of the characteristics of the original signal. The artificial inter-arrival timing is noise that is added to the signal. In order to reduce the effect of this noise, a more natural signal representation, shown in the bottom of Figure 2.7 and described in the following section, is used.

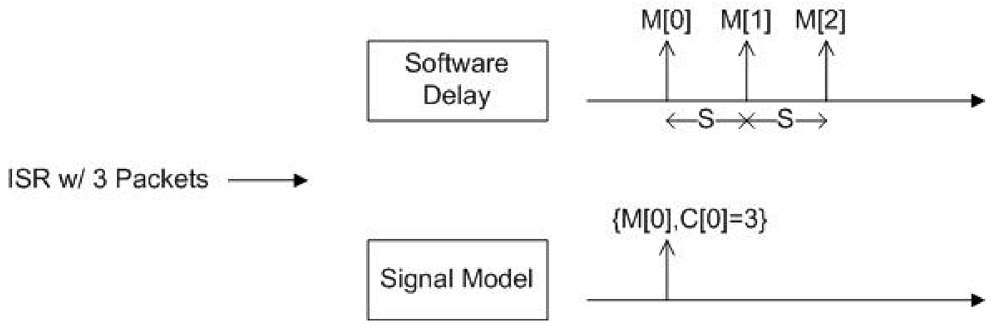

Figure 2.7: Example of Software Induced Delay vs. Signal Model

2.2.4 Signal Representation - SigVec

Some alterations to the signal representation become necessary when the effects of the software measurement system are taken into account. In particular, the problem outlined in Section 2.2.3, where system processing being done for each packet creates an inter-arrival time between packets in the same coalesced group that is near uniformly equal to S seconds. It is desirable to remove the measurements that are affected by the software processing delay, and to do so we propose our signal representation, which we have termed *SigVec*. In *SigVec*, the signal is represented by the vector, $X(k)$:

$$X(k) = \{M(k), C(k)\} \tag{2.1}$$

Where $M[k]$ represents the timestamp generated by the system for the first packet in the coalesced group, and $C[k]$ gives the number of packets received in that group. The number of packets received in each group, $C[k]$, is recorded because this information is needed by many statistical analysis methods. As a simple example, computing the average rate or inter-arrival time between packet arrivals in a computer network requires the number of packets contained in each measurement. Let λ denote the average inter-arrival time between packets, and let λ' denote the average inter-arrival time between *measurements* recorded on a software measurement system using our signal representation. Then the average inter-arrival times are related by the expression $\lambda' = E[C[k]] \cdot \lambda$, i.e., the average inter-arrival between measurements is equal to the average inter-arrival time between packets multiplied by the average number of packets per measurement. The information contained in $C[k]$ is also required for more sophisticated statistical analysis, in particular it is important for deriving an estimate of the renewal density, which we discuss in Chapter 3 and use in our anomaly detection method.

SigVec captures precisely the measured information: each interrupt provides accurate timing information for one packet and a packet count. Depending on the type of interrupt coalescence used in the system the timing information may indicate the actual timing of the last packet or the first packet in the measurement. This is the case in systems that use PIC. As an example, in Figure 2.8, the measurement timestamp, $M[1]$, generated for the first serviced packet is equal to $M[1] = m[3] + 1/r + S$.

Here $m[3]$ indicates the time when third packet has been transferred to the system via the DMA channel. In this example we assume that $C_{fix} = 3$ for PIC, so $m[3]$ is the last packet in the group of packets. Therefore the timestamp of $M[1]$ indicates the time when the last packet in the group of packets has been transfered to the system, plus the ISR handling time $1/r$ and the system processing time S. Assuming $1/r$ and S are constant, the final measurement timestamp, $M[1]$ indicates the actual packet timing of the last packet in the measurement with a constant shift of $1/r + S$. The constant shift is the same for every measured timestamp, and thus can be ignored.

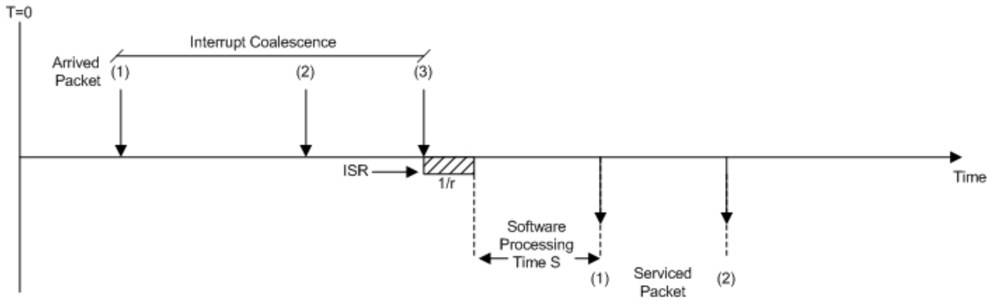

Figure 2.8: Measurement Timestamp for PIC

A similar situation arises in systems that use TIC. As an example, in Figure 2.9, the timestamp $M[1]$ is equal to $M[1] = m(1) + T_{fix} + 1/r + S$. In TIC the timer begins counting down after the initial packet, and the interrupt is asserted once the timer has reached zero. Thus the timestamp generated for the measurement, $M[1]$ is the timing of the first packet in the group plus a constant shift equal to $T_{fix} + 1/r + S$. While knowing that the measurement timestamp indicates the first or possibly the last packet in the group does not affect the analysis techniques, it is important to note that only one packet in each coalesced group does have accurate timing, and thus the signal representation is appropriate.

The key point regarding the signal representation is that by grouping all packets received during the same interrupt into a single measurement the error associated with

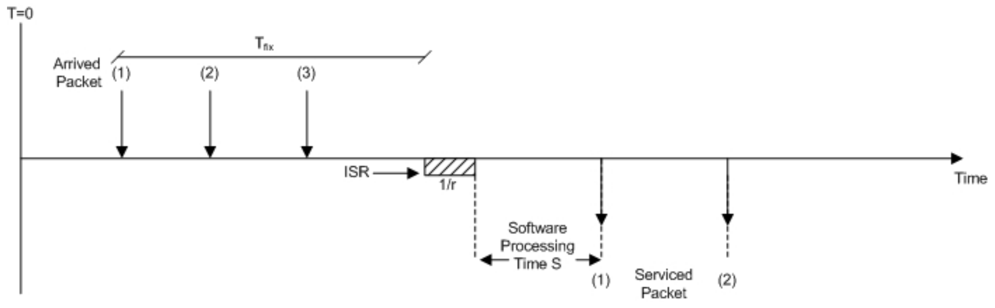

Figure 2.9: Measurement Timestamp for TIC

software processing is mitigated, i.e., the artificial S second inter-arrival time is removed. Further, instead of being viewed as a timing distortion, interrupt coalescence can be seen as a subsampling mechanism that reduces the number of measurements that can be used for analysis. Thus only the timing delay from transferring the packet to memory remains. Note that when not measuring the time at the beginning of a packet, and instead generating a timestamp indicating the end of the packet, the delay due to data transfer actually does not alter the relevant timing statistics. In other words, the average inter-arrival time between packets remains the same regardless of whether the timestamp reflects the beginning or the end of a packet.

Therefore the process of generating measurements using a software based measurement system is equivalent to methods which sample only a subset of incoming packets, such as those described in [4, 16, 78]. The key difference is that in the mentioned works subsampling is a process applied to the measured data by the software or hardware collecting the measurements. Instead, in a software measurement system subsampling is controlled by the operation of interrupt coalescence, which is primarily designed to reduce the number of ISRs asserted by the NIC and is not designed for generating measurements. In particular there is no reason to expect that these measurements preserve information that is important for Internet traffic analysis. In the next section a general model for interrupt coalescence will be introduced, which we also show how to optimize. The general model for interrupt coalescence, is event counts, and can be used to generate many of the subsampling strategies described in [16], and any others that involve event counts. Not that subsampling strategies that use timing between events cannot be implemented. We do not use timing between events because as compared to counters measuring timing between events, their implementation is more difficult in a measurement system, particularly in lower cost systems.

2.3 Subsampling Optimization

One main contribution of this thesis is to present for the first time a formal approach to designing subsampling methods, so that measurements are optimized to improve Internet analysis. We are particularly concerned with quality-of-service monitoring tasks such as end-to-end link capacity estimation, available bandwidth estimation, and bottleneck detection, as well as security applications such as denial-of-service (DoS) attack detection. Besides being important for proper network operation, all these tasks share a common characteristic of requiring sufficiently accurate inter-arrival timing data between packets. Given the tremendous packet rates associated with Internet network traffic, combined with the subsampling architecture already built-in to standard measurement systems, optimizing subsampling for Internet analysis is extremely useful.

Our work in optimizing subsampling, which was first presented in [53], was done considering interrupt coalescence as the method of subsampling. The various practical IC techniques that have been proposed [37], typically use simple timers triggered by packet arrivals. The focus in these designs has been on maintaining reasonably low interrupt rates, and no formal consideration has been given to how groupings should be optimized to facilitate relevant Internet traffic analysis tasks.

We note that while the work here focuses on interrupt coalescence in software based measurement systems, and therefore only considers packet arrival measurements, the techniques derived are general and can be applied to many other scenarios, such as those discussed in Section 2.4. For example, instead of being used to implement interrupt coalescence, subsampling can be applied in high-rate (hardware) based measurement systems to reduce the storage cost required for storing long segments of Internet traffic. Subsampling of high-rate measurements can be done using the existing PSAMP/IPFIX protocols [2,3]. These protocols define the general operation of measurement systems, yet allow the specific operation of the systems to be left to the user or system designer. Thus, optimized subsampling methods could be used in measurement systems under the PSAMP/IPFIX protocols. Further, subsampling is also applied to flow based measurements, such as NetFlow, therefore our optimization techniques could also be applied to flow based measurements as well.

The starting point for our work is that for various analysis tasks, *measurement systems should preserve both first and higher order inter-arrival distributions*, where the order refers to how many packets are received in between the measured timestamps. Example analysis applications include, end-to-end link capacity and available bandwidth estimation [41,61], and our anomaly detection method, to be described in Section 3.2.3, that computes an estimate of the renewal density. Thus, we propose that various IC (and subsampling) approaches can be compared by obtaining first and higher-order inter-arrival distributions and quantifying, e.g., using the Kullback-Leibler divergence, how different they are from those that would be obtained from the original data, for which timing is accurate for all packets as recorded by a high-rate measurement system.

Our proposed optimization techniques are based on two main observations. First, existing IC methods are based on timers triggered by packet arrivals. This leads to potential biases in estimated inter-arrival times, since different packet inter-arrival patterns affect the timers, which in turn are used to generate measurements. Thus we propose measurement techniques that group packets to generate interrupts *independently of their inter-arrival times.* Specifically, before each measurement, the system decides how many packets should be aggregated into one measurement and waits as long as needed for the required number of packets to be gathered before generating an interrupt. Second, such measurement systems are completely specified by the distribution of number of packets per measurement.

There are many ways of selecting the distribution of the number of packets per measurements. Since we consider applications that use different inter-arrival orders, we propose the following two metrics to that we use to select distributions of packets per measurement that are optimized for particular Internet traffic analysis applications. The first metric aims at maximizing the number of accurate first and higher order measurements, and is optimized for tasks where smooth approximations of inter-arrival times are required. The second metric aims at distributing more uniformly measurements across inter-arrivals of different orders, and it is well suited for anomaly detection tasks. While we only consider the two previously mentioned Internet traffic analysis applications and corresponding metrics, for other analysis applications different metrics can be designed and used to select the optimal distribution of packets per measurement for that particular task.

2.3.1 Problem Formulation

Using the signal representation described in Section 2.2.4, denote the "ideal" measured signal to be $X'(k) = \{M'(k), 1\}$, where exact timing is available for all packets.

If $X'(k)$ is available then the m-th order inter-arrival, $J'_m(k)$, for measurement $X'(k)$, is obtained by taking the difference $J'_m(k) = M'(k) - M'(k-m)$. For a given measured signal, $X(k) = \{M(k), C(k)\}$, produced by an IC measurement system the m-th order inter-arrival, $J_m(k)$, is given by the difference, $J_m(k) = M(k) - M(k-m)$ where m indicates the sum of the number of packets in the j previous measurements, as in the following equation (note that $C(k-j)$ is not included):

$$\sum_{i=0}^{j-1} C(k-i) = m \tag{2.2}$$

The inter-arrival histogram at order m is computed using all of the inter-arrival measurements of order m. Without subsampling, every measurement $X'(k)$ produces an inter-arrival at every order m, however, when IC or subsampling is used then, for a measurement $X(k)$, the inter-arrival times that can be computed depend on the $C(k)$'s. As stated, our goal is to optimize the coalescing of packets in a way that maximally retains the first and higher order inter-arrival timing information. Our goal is in contrast with methods like TIC, PIC and HIC used in [37]. These methods coalesce packets in order to maintain a desired interrupt rate, and not to preserve any timing information. Further, these methods use techniques that can potentially bias the inter-arrival timing information. For example, HIC generates measurements when a timer counts down between packets biasing inter-arrival time estimates because, for example, $C(k) = 1$ *only if* $M(k) - M(k-1)$ is large (in order for the timer to expire).

To preserve timing information, and eliminate bias, our IC mechanism selects how many packets are aggregated into each measurement independently of their arrival times. Instead, our method randomly selects the number of packets in each measurement following an independent and identically distributed probability mass function (pmf) that is optimized based on the criteria mentioned earlier. Let l denote the number of packets included in the next measurement. Then l is a random variable with pmf $\{p(1), p(2), ..., p(N-1), p(N)\}$, $\sum_{l=1}^{N} p(l) = 1$. Thus, the k-th measurement includes l packets with probability $Pr(C(k) = l) = p(l)$.

Note that with this implementation the pmf of l completely defines how the system operates. By denoting $P(N)$ to be the set of all possible pmfs with a maximum of N packets per measurement, our goal will be to define metrics that apply for various analysis tasks which we can optimize to find the best pmf in $P(N)$ for a given analysis task.

2.3.2 Optimization Techniques

As a starting point we define $I(n)$ to be the percentage of inter-arrival measurements retained at order n. $I(n)$ is the ratio of the number of inter-arrival measurements of order n obtained using the subsampled signal X. The number of inter-arrival measurements of order n obtained using the ideal signal X'. These are the inter-arrivals of order n that can be directly derived from measured data, i.e., without using histograms derived for lower order arrivals to estimate higher order inter-arrivals. $I(n)$ is a function of the pmf of the number of packets per measurement. For example, if $p(2) = 1$ and $p(1) = p(3) = ... = p(N) = 0$, i.e., all measurements have two packets, then $I(2 \cdot i + 1) = 0$ and $I(2 \cdot i) = 1$ because it is not possible to get an inter-arrival order that is an odd number if all measurements have two packets.

$I(n)$ is computed by taking the summation of all possible ways to get an inter-arrival of order n (as computed using (2.2)), this is equivalent to finding the integer partitions of n. A partition of n is a set of strictly positive integers that sum to n. Let $s_n(j)$ be a partition of n with j elements, i.e., $s_n(j) = \{e_1, e_2, ..., e_{j-1}, e_j\}$ and $\sum_{i=1}^{j} e_i = n$. Let $S(n)$ be the set of all partitions of the integer n. Computing partitions is done recursively using a method such as that of Wilf, which computes the partitions in Gray Code or "minimum change" order [59]. Wilf approach and other related methods list partitions in non-increasing order. For example, a partition is listed only as $\{3,1,1\}$, yet it can also be reordered as $\{1,3,1\}$ and $\{1,1,3\}$. When computing $I(n)$ all reorderings are considered different, i.e., each reordering adds to the number of inter-arrival measurements, at a particular order, retained following subsampling. Thus, for a given partition, $s_n(j)$ with j elements, the number of distinct permutations is given by $\frac{j!}{j_r(1)!\cdot j_r(2)!\ldots j_r(i)!}$ where $j_r(i)$ is the number of elements e_j in the partition equal to a specific integer $r(i)$. In our example, for the partition $\{3,1,1\}$ we have $j=3$ elements, $j_1=2$ and $j_3=1$ giving $\frac{3!}{2!\cdot 1!}=3$ distinct permutations.

To compute the function $I(n)$ we define the mapping α which takes each element in the partition as the index into the pmf, then computes the product of these elements:

$$\begin{gathered}\alpha : S(n) \longmapsto P(N) \\ \alpha(s_n(j)) = \alpha(\{e_1, e_2, ..., e_j\}) = p(e_1)\cdot p(e_2)\cdots p(e_j)\end{gathered} \tag{2.3}$$

Here we restrict our partitions to be strictly less than the maximum number of packets per measurement given our pmf, i.e., $n < N$.

Finally, the function $I(n)$ is computed by:

$$I(n) = \sum_{s_n(j)\in S(n)} \frac{j!}{j_1!\cdot j_2!\cdots j_r!}\alpha(s_n(j)) \tag{2.4}$$

multiplying each partition by its number of distinct permutations, and summing over all possible partitions of n.

For example, the only way to get an inter-arrival of order 1 is to have a measurement with one packet, thus $I(1) = p(1)$. For $I(2)$ there are two partitions $\{2\}$ and $\{1,1\}$, so $I(2) = p(2) + p(1)^2$ Finally, for $I(3)$ there are three partitions $\{3\}$,

$\{2,1\}$ (this partition has two distinct permutations $\{2,1\}$ and $\{1,2\}$), $\{1,1,1\}$ and $I(3) = p(3) + 2 \cdot p(2) \cdot p(1) + p(1)^3$.

We now propose two different metrics to be optimized with our choice of pmf.

Formulation 1 *OICv1 - Maximize the total number of inter-arrival measurements of multiple orders, noting that each measurement $X(k)$ contributes to multiple inter-arrival measurements. Formally:*

$$\begin{aligned} &\max_{\{p(i)\}\in P(N)} \sum_{i=1}^{N} I(i) \\ &s.t. \sum_{i=1}^{N} i \cdot p(i) = \hat{m},\ \sum_{i=1}^{N} p(i) = 1 \end{aligned} \tag{2.5}$$

which maximizes the percentage of inter-arrival measurements retained, up to order N, subject to the constraints that the sum of the pmf is unity and the average value is a user defined $\hat{m}$ that gives the desired reduction in measurement rate.

The optimization is carried out using a sequential quadratic programming approach to solve the constrained, nonlinear, multi-variable formulation via an algorithm adapted from [31]. This optimization, which we term OICv1, results in a pmf we characterize as "on/off", where a significant probability weight is given to $p(1)$, and the remaining weight given to a much higher order to achieve the desired average value. The "on/off" characteristic of the pmf is caused by the fact that $I(n)$ can be computed for any chosen pmf, and each pmf resulting potentially resulting in a very different percentage of inter-arrival times preserved at each order. This leads to a trade-off, where it may be possible to increase the number of measurements at a given order, at the expense of decreasing the number of measurements at a different order. Note that if the desire was for the number of measurements as so other order

k to be maximized, i.e., if we want $I(k)$ to be large, then giving significant weight to $p(1)$ is not the best strategy.

For example, optimizing the pmf for an average of 6 packets per measurement, and setting $N = 40$ leads to a solution of $p(1) = .8214$ and $p(29) = .1786$. The large value of $p(1)$ helps this method achieve the maximum number of inter-arrival measurements. To illustrate, consider Figure 2.10, which shows the inter-arrival measurements retained using OICv1. Because $p(1)$ is large OICv1 produces many consecutive single packet measurements, and a few measurements with many packets. When n single packet measurements occur consecutively they produce $n-1$ first-order inter-arrivals, $n-2$ second-order, etc. In fact, using (2.2) we see that the number of inter-arrivals retained at order $n < 29$ is equal to $p(1)^n$; in other words it drops of geometrically. Alternatively, consider the case where all measurements have 6 packets. This formulation produces the same average number of packets per measurement, however, for n measurements this formulation only produces $n-1$ sixth-order inter-arrivals, $n-2$ 12th-order, etc., significantly less total inter-arrivals than the previous case.

The relationship between the value of N and the "off" probability weight, i.e., $p(29) = .1768$ in the above example, can also be seen in Figure 2.10. The formulation in (2.5) maximizes the total number of inter-arrival measurements, which in the figure is equivalent to maximizing the area under the inter-arrivals retained curve. Notice that the "off" probability weight is not given to $p(N)$. While it might seem intuitive that to maximize the number of measurements one should maximize the value of $p(1)$. However, maximizing the value of $p(1)$ would require that the "off" probability order $p(n)$ be set at a larger n to maintain the same expected value, i.e., $p(1) + n \cdot p(n) = \hat{m}$. Since we are trying to maximize the area under the curve, selecting a smaller order n for the "off" probability is actually better.

For analysis the problem with OICv1 lies in the fact that the percentage of inter-arrival measurements retained decreases geometrically in $p(1)$, as shown in Figure 2.10. Thus, as the inter-arrival order increases, from $1-29$ in our example, we have very few measurements available. To solve this problem one approach is to use the inter-arrival data from the lower orders, specifically the first order data, to estimate the inter-arrival data of the higher orders. This could be done by: estimating the probability distribution of the first order inter-arrival times (using measurements containing only one packet), and using the estimated probability distribution to estimate the probability distribution of higher order inter-arrivals by taking the convolution of the first order inter-arrival distribution with itself multiple times. However, estimating the higher order inter-arrival distributions this way leads to smooth higher order inter-arrival estimates due to the convolution operation. The problem with the smooth estimate is that it will miss potentially interesting anomalous events that occur infrequently. For example, if many "normal" events occur between anomalous events, then the inter-arrival timing between consecutive anomalous events can only be observed in the actual higher order inter-arrival time distribution and not in the estimated distribution.

To solve this problem we propose a second approach, more suited for anomaly detection, based on a different optimization metric that distributes measurements more uniformly across orders. The second formulation provides better estimates of higher order inter-arrival time distributions based strictly on measured data, i.e., without using lower order inter-arrival distributions to estimate higher order ones.

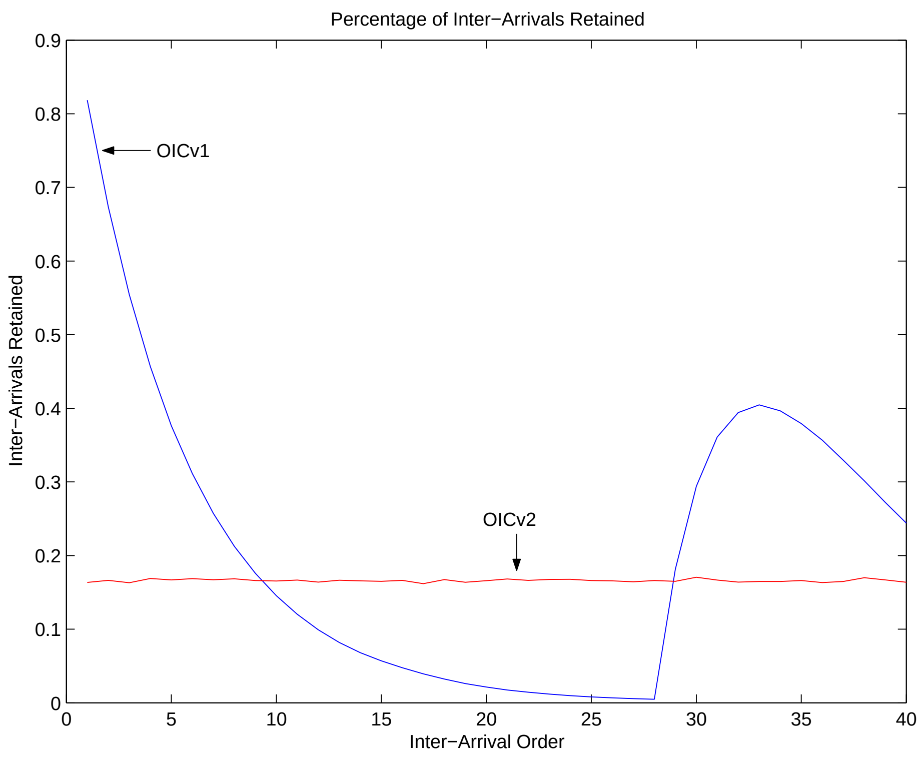

Figure 2.10: Inter-arrivals retained using OICv1 and OICv2 for $\hat{m} = 6$

Formulation 2 *OICv2 - Find the pmf that produces a uniform distribution of $I(n)$ for all n.*

$$\begin{aligned} &\min_{\{p(i)\}\in P(N)} \sum_{i=1}^{N} (q - I(i))^2 \\ &s.t. \sum_{i=1}^{N} i \cdot p(i) = \hat{m}, \sum_{i=1}^{N} p(i) = 1 \end{aligned} \tag{2.6}$$

which minimizes the error between a uniform distribution and the $I(n)$ functions, under the same constraints as the first optimization.

The difficulty with this problem is the selection of the parameter q, which is merely an arbitrary value that the optimization attempts to make all orders of $I(n)$ equal to. In general this optimization can be solved for any given value of $q \leq N$. However, solving the problem this way only leads to the minimum error solution, i.e.,

$\sum (q - I(n))^2$, for that q. Instead, by relaxing the condition $\sum_{i=1}^{N} i \cdot p(i) = \hat{m}$, we can find an exact solution such that $\sum (q - I(n))^2 = 0$. For the exact solution the value of q is not arbitrary, and is instead related to the expected value, $\hat{m}$.

First we derive the $I(i)$ and $p(i)$ as a function of q. Starting with $I(1) = p(1)$ and assuming an exact solution exists $I(1) - q = p(1) - q = 0$ (from (2.6)), thus $p(1) = q$. The remaining $p(i)$ are solved for iteratively as follows:

$$\begin{gathered} I(2) - q = p(2) + p(1)^2 - q = 0 \;\Rightarrow\; p(2) = (1-q) \cdot q \\ I(3) - q = q = p(3) + 2 \cdot p(1) \cdot p(2) + p(1)^3 - q = 0 \;\Rightarrow\; p(3) = (1-q)^2 \cdot q \\ \ldots \\ I(n) - q = 0 \;\Rightarrow\; p(n) = (1-q)^{(n-1)} \cdot q \end{gathered} \tag{2.7}$$

The general solution, $p(n) = (1-q)^{(n-1)} \cdot q$, can easily be recognized as the probability of success on the n-th attempt given a geometric distribution. Therefore the geometric distribution, with parameter q, gives an exact solution to the second optimization formulation. Further, the expected value of the geometric distribution is $1/q$, therefore using the relation $1/q = \hat{m}$ we can choose $q = 1/\hat{m}$ to meet the constraints of 2.6.

Finally, we claim that value of $q = 1/\hat{m}$ is the *maximum* value that solves the formulation exactly. To show this, assume there exists another value, say $q' > q$, which solves this formulation exactly. As shown above, this implies that $p'(n) = (1-q')^{(n-1)} \cdot q'$. Since $p(n)$ is given by a geometric distribution, then $\sum_{i=1}^{N} p(i) = (1-q)^{(n-1)} \cdot q = 1$. However, for $p'(n)$, the sum $\sum_{i=1}^{N} p'(i) = (1-q')^{(n-1)} \cdot q' > 1$ because $(1-q')^{(n-1)} \cdot q' > (1-q)^{(n-1)} \cdot q \;\; \forall n$. Thus, the new solution is not a valid pmf, and the solution $q = 1/\hat{m}$ is the *maximum* value that solves the formulation exactly. Selecting q to be the maximum value that solves the formulation exactly is not an explicit constraint of our formulation, it is however implied. Since a geometric distribution of the pmf is the only one that solves the formulation exactly, we could select the pmf to be geometrically distributed with parameter $r < q$. The expected value of this new pmf would be $1/r > 1/q$ and the expected number of packets per measurement, i.e., the subsampling rate, would be too large and the constraints of our optimization framework would not be met. Further, by maximize the value of q we actually maximize the total number of inter-arrival measurements retained while meeting the constraints of our formulation. This occurs because the number of inter-arrival measurements retained is equal to $\sum I(i)\ i \in [1, N]$, but $I(i) = q\ \forall i$. Therefore, the sum, $\sum I(i)\ i \in [1, N]$, is actually equal to $N \cdot q$ and is maximized by maximizing the value of q.

It is important to notice that the solution to the second optimization is equivalent to uniform random sampling. In uniform random sampling each incoming event is either retained, i.e., sampled, with probability q or skipped with probability $1 - q$. This equivalence is important because multiple theories regarding thinning a renewal process, which will be discussed in Section 3.2.1.1, are based on uniform random sampling of a renewal process. Thus, there is a strong link between thinning a renewal process and our proposed methods for subsampling Internet measurements.

2.3.3 Performance Evaluation Using Synthetic Data

Since our goal was to retain as much inter-arrival information as possible following coalescence, we select the Kullback-Leibler (KL) divergence as a metric to compare the performance of our optimizations with existing IC methods. We compare the performance of our optimized IC methods, OICv1 and OICv2 (where v1 represents the method derived from (2.5) and v2 from (2.6)), versus three common IC strategies: fixed pack IC (PIC), fixed time IC (TIC), and the hybrid IC (HIC) method developed by Intel [37].

The KL divergence is computed by generating histograms to estimate the inter-arrival timing distribution using ideal measurements (where each packet receives an individual timestamp) and measurements generated by simulating interrupt coalescence with a measurement subsampling rate of 6 (input/output measurement rate). For each signal we compute the histogram for inter-arrival orders up to 40, and measure the divergence between the inter-arrival distribution estimates at each order.

While not a true distance metric, KL divergence is a common measure of the difference between two probability distributions [45]. The divergence, $D_{KL}(P||Q)$, between distributions P and Q is computed using:

$$D_{KL}(P||Q) = \sum_{i} P(i) \ln \frac{P(i)}{Q(i)} \tag{2.8}$$

The KL divergence is an average of the difference (logarithmic difference) between the two distributions [45]. Other divergence or distance metrics could be used, such as the total variation distance, which finds the greatest distance between the probability distributions P and Q. We use the KL divergence here because the distributions we encounter in simulation are smooth, therefore, a greatest distance metric (such as the total variation distance) is not as meaningful as an average difference metric (the KL divergence) because the difference between two smooth distributions is likely small everywhere and the greatest distance would vary randomly to different points in the distribution. Even in practice, when we estimate the distributions of inter-arrival times from Internet traffic, the distributions are smooth, yet have small random variations around this smooth curve because inter-arrival times between packet arrivals in Internet traffic do not follow an ideal random process. Thus, we select the KL divergence to give an overall measure of the difference between the distributions.

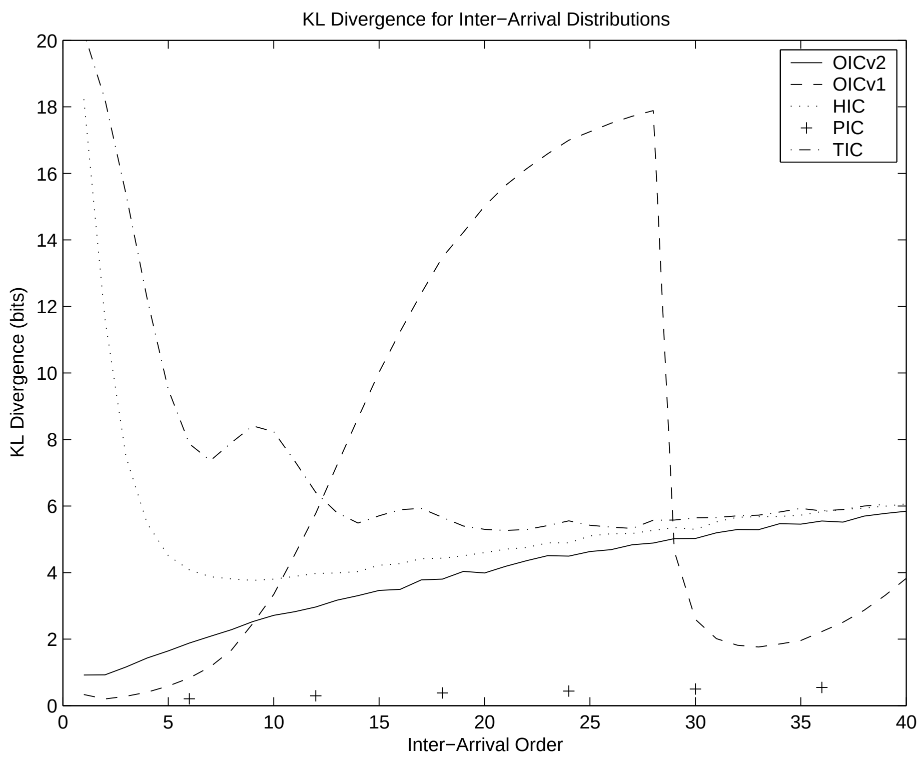

Figure 2.11: KL divergence of inter-arrival distributions using only received data

In the following analysis we simulate Internet traffic with exponentially distributed inter-arrival times. While it has been shown [58] that a Poisson process does not accurately model Internet traffic we use it here to show our solution applies to a more general class of signals, i.e., signals where packet size does not constrain minimum inter-arrival times. For actual Internet network traffic, the performance of our method and the IC methods we use for comparison will be different than in the Poisson case. We will discuss the important performance differences of the methods when using Poisson and Internet traffic following the Poisson discussion in Section 2.3.3.1.

Figure 2.11 shows the KL divergence measured using only the inter-arrival times computed using the actual measurement data. The plot for PIC only contains points at multiples of the subsampling rate, for example r, because for all measurements PIC generates $C(k) = r$. Because PIC generates measurements with identical numbers of

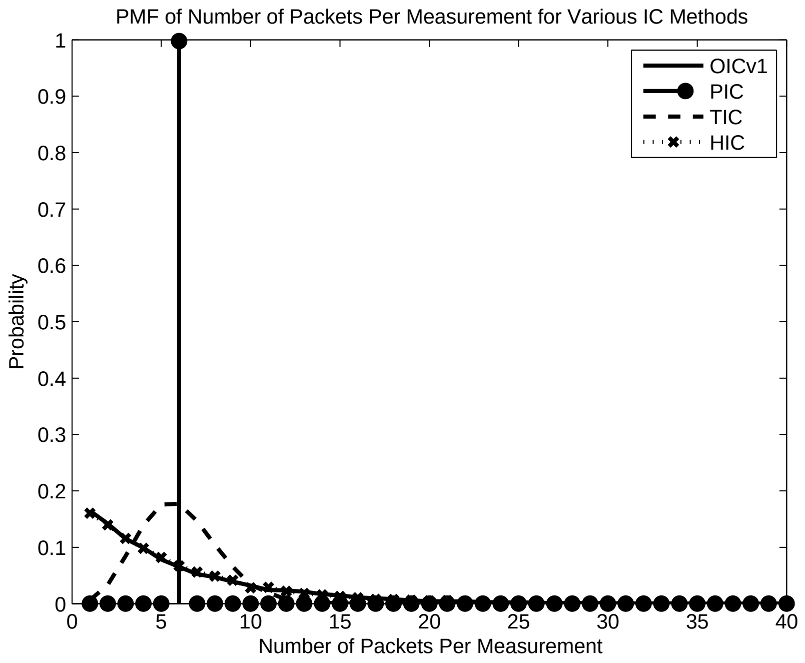

Figure 2.12: PMF of the number of packets per measurement for the various IC methods

packets per measurement it contains no inter-arrival information for orders that are not a multiple of r.

For TIC the KL divergence plot contains multiple bumps, where low points (when the divergence is minimum) occur near multiples of the subsampling rate 6. This happens because the value of T_{fix} used in TIC was selected in order to achieve the subsampling rate of 6. Selecting T_{fix} this way creates a distribution of packets per measurement that is centered around 6, which is shown in Figure 2.12. Much like the PIC case where all measurements have 6 packets, the KL divergence for TIC is minimum near multiples of the subsampling rate.

The distribution of the number of packets per measurement for HIC and OICv2, as shown in Figure 2.12, is nearly identical, and because of this we would expect that the KL divergence for the two methods to be similar as well. However, based on the result shown in Figure 2.11 this is clearly not the case. The reason for this is due to the way HIC operates, in particular the packet timer prevents measurements from having an inter-arrival time shorter than T_{pack}. Therefore, the first and low order measurement inter-arrival time distributions are skewed towards higher inter-arrival times, which is quite different from the ideal measurements and the KL divergence at low orders is very large. The skew in the distribution impacts the inter-arrival distribution up to order 6 because this is the value that the timers T_{pack} and T_{abs} were designed for. Beyond order 6 HIC performs similarly to OICv2, which fares the best overall using only the data received to generate the inter-arrival histograms.

The limitation of OICv1, the on/off measurement behavior, is apparent from the Figure 2.11 as well (OICv1, is not shown in Figure 2.12, but its distribution would be bimodal with $p(1) = .8124$ and $p(29) = .1786$). The KL divergence is very small at low orders, and for inter-arrival orders near the 'off' probability value ($p(29)$ in this example), which is caused by the combination of 'on' and 'off' measurements generating inter-arrival times with orders close to 1 and 29 in this case. While OICv1 performs well at these orders we see that the divergence increases rapidly at inter-arrival orders in between the 'on' and 'off' probability values and orders much larger than the 'off' value.

OICv1 generates the largest number of total inter-arrival measurement by generating a majority of single packet measurements. This allows OICv1 to accurately replicate the first order inter-arrival time histogram. We exploit the accurate first order inter-arrival timing information to estimate the higher-order histograms by taking the convolution of the first-order histogram multiple times, an operation we describe in detail in Section 3.3.3. With this modification OICv1 can achieve very low values of KL divergence even at high inter-arrival orders, however, as mentioned before, the alternative method to generate the higher order inter-arrival histograms creates smooth approximations of the higher order distributions that are not practical for

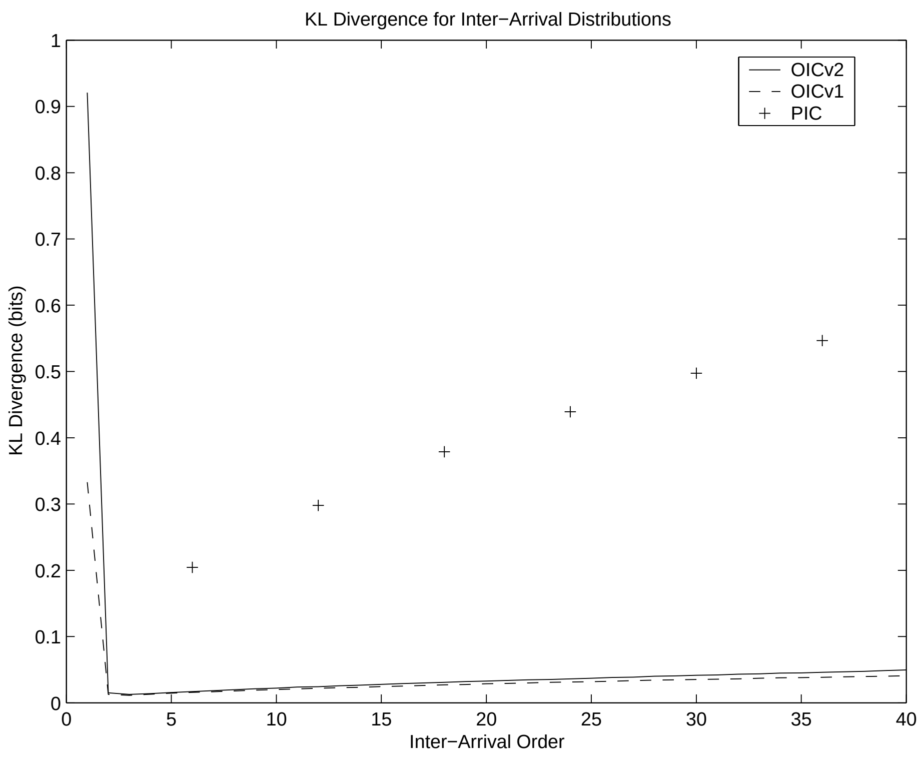

Figure 2.13: KL divergence of inter-arrival distributions using estimates from low order data

detecting anomalous traffic.

Figure 2.13 shows the KL divergence performance using estimates of the high order inter-arrival histograms obtained from the lower order inter-arrival data. From the figure we see that OICv1 shows the smallest KL divergence of any IC method. For some methods the KL divergence actually increases, as is the case for HIC and TIC which are not pictured because their divergence is very large. This is caused because certain methods (like HIC) are prone to biases in measurements with low numbers of packets per measurement. Because HIC only generates a measurement with one packet if the inter-arrival time between consecutive packets exceeds a given time the first-order inter-arrival histogram is biased towards long inter-arrival times and is a poor indication of the actual distribution of inter-arrival times.

2.3.3.1 Internet Traffic Measurements

As mentioned previously the performance of our optimized IC techniques and the standard IC methods differ when actual Internet traffic is used in place of Poisson traffic. One area where Internet traffic deviates from Poisson is in what is called "burstiness" of Internet traffic. Burstiness of Internet traffic manifests as short bursts of many packet arrivals, followed by relatively long periods of inactivity. An example of this behavior would be when a user browses on to a website: their web client requests the webpage from the server, which requires a significant amount of data to be transmitted from the server to the client. Once the website is loaded the user will typically spend a large amount of time, relatively speaking, reading the webpage before browsing on to another website. Aggregate Internet traffic consists of many such bursty sessions, combined with more regular traffic, e.g., video streaming, where the traffic rate is more constant. In terms of performance, the difference between our optimized IC methods and the standard methods, HIC and TIC for example, is that our methods are signal independent. Signal dependence or independence is important when trying to measure traffic characteristics. For example, during bursty periods of Internet traffic HIC will likely coalesce many packets together because the packet timer does not count down to zero when packets are arriving quickly. Conversely, when using HIC any stray packets that arrive during the non-bursty, or inactive, period will not be coalesced with other packets. Therefore, the measurements containing only a single packet will only be from the inactive periods. The signal dependent behavior of HIC can influence measures of traffic characteristics. Alternatively, when using our optimized methods the chance that a measurement contains only a single packet does not depend on whether the traffic was in a bursty or inactive period. Thus, because our optimized methods generate measurements independently of the incoming traffic, our methods can produce more reliable estimates of traffic characteristics than the standard IC techniques.

2.4 Generalization to High Rate Measurement Systems

One main benefit of the signal representation we selected in Section 2.2.4 is that it is completely general and can be used for measurements from *all* types of existing measurement systems. This is particularly convenient considering the vast selection of measurement system architectures, which vary in terms of system cost, accuracy and measurement type. Similarly, our method to optimize interrupt coalescence introduced in Section 2.3 is directly applicable to measurements from *all* measurement systems that employ our signal representation.

For example, as mentioned in the introduction, hardware based systems are capable of producing very accurate timestamps for individual packet arrivals in real-time. Examples of such systems include CoralReef, developed and maintained by CAIDA [43], or the commercial line of DAG network interface cards developed by Endace [23, 24]. While these systems do not require subsampling of measurements, in certain scenarios subsampling might be desirable to reduce data storage costs. For hardware based systems our *SigVec* representation is a suitable method to record measurements whether or not subsampling is used.

Further, our signal representation also applies to measurement systems that do not record packet arrivals, like NetFlow developed by Cisco, which records measurements of network flows [15]. In systems that record network flows the start of the flow is typically recorded, thus a timestamp representing the flow start could be reflected in our signal representation. Subsampling of flows is also common in analysis to reduce the computational cost as was done in [26]. Further, in detection systems that search for long-range periodicities in flow arrivals, such as the work by Bartlett, et. al. [7], subsampling could be potentially be applied and our signal representation used.

Our signal representation and methods to optimize subsampling are particularly relevant given the recent protocols defined by the IETF PSAMP and IPFIX working groups [2, 3] for creating a flexible distributed measurement system architecture. The PSAMP and IPFIX working groups defined a set of common protocols to be used in certified measurement nodes (called recording nodes), which are designed for recording both packet and flow based Internet traffic events. Because the systems are designed to be deployed in a distributed manner one desirable quality of the system is low-cost. In order to make such systems low-cost both protocols define what they call selection processes, which are methods to select with incoming event to sample. In other words, a selection process is a method to subsample the incoming events, and because the selection process is general, a subsampling strategy designed using our optimization techniques could be implemented. Further, the distributed architecture is designed such that the measurements are sent from the recording nodes, via an exporting process, to a collector node where analysis would take place. Subsampling is an effective method to reduce the data transmission cost between recording and processing node, which is something we consider in Chapter 4.

Thus, our signal representation *SigVec* and methods to optimize interrupt coalescence can be generalized to many existing measurement systems.

2.5 Conclusion

In this chapter we analyzed and modeled the timing errors inherent in software based measurement system. This modeling was necessary in order to make such systems, which are desirable for their low-cost, useful for general Internet traffic analysis tasks. From this system modeling we proposed a signal representation, called *SigVec*, which represents the accurate timing information following the software system processing. In a software system using interrupt coalescence accurate timing is available for only one packet per group of packets that are serviced during the same interrupt service request. Because we record only one accurate timestamp for the group of packets the operation of interrupt coalescence controls the subsampling of the measured events. To optimize the operation of interrupt coalescence, we created a metric that can be used to produce software system measurements optimized for specific Internet traffic analysis tasks. By optimizing generation of measurements and removing their dependence on packet arrivals we can improve estimates of inter-arrival distributions, which we use in subsequent chapters to perform Internet traffic analysis.

Finally, the signal representation and method to optimize interrupt coalescence are general enough to be applied to most existing measurement systems, including high rate measurement systems, such as the previously mentioned hardware based systems, and systems that record measurements of flows instead of packet arrivals.

Chapter 3:

Anomaly Detection with the IA^{2}D Detection System

3.1 Introduction

As shown in Chapter 2, Internet traffic events, and more specifically packet arrivals, occur as discrete events, with each event occurring at a unique time. Thus a collection of packet arrivals can be seen as a realization of a point process, where each event (e.g., the n-th packet arrival) is indicated by the time it occurred, t_n. Similarly, traces recorded using Internet measurement systems, e.g. [24], produce a time stamp, say t'_n for each packet measured (in the case of a measurement system this time could indicate the actual packet arrival time plus some delay due to system processing).

Although the point process is a natural representation of packet arrivals and network measurements, it is common in network traffic analysis to convert the collection of measurements to a time series representation by uniformly sampling the signal, as described for example in [18]. The time series representation is generally favored over the point process representation in tasks that analyze the long term characteristics of Internet traffic. This is done because, for a uniformly sampled time series, signal processing tools can be used for analysis, which can efficiently determine long term characteristics in time series. For example in [6, 12, 17, 18, 41] a uniformly sampled signal representation enables the use of standard signal processing tools such as wavelets [6, 41], and spectral analysis [17, 18], which have been shown to be effective methods to detect network performance problems like bottleneck links [41] and network anomalies [6].

The point process representation has been used previously to analyze Internet traffic, for example in [49] and [42], however in this work only the inter-arrival time between consecutive events was analyzed. Computing the inter-arrival time between consecutive events is useful for determining some traffic characteristics, such as average rate, but looking only at consecutive events fails to capture the long range behavior in Internet traffic. In this chapter we present a number of formulations, with existing foundations in renewal theory, that we use to characterize the long term behavior of Internet traffic measurements while maintaining the timing accuracy inherent to the point process representation. We introduce these formulations, called the renewal function and renewal density, that allow us to perform similar network analysis tasks as were done with the time series representation, i.e., detecting bottleneck links and network anomalies, without having to convert from the point process representation of the measurement system to the time series representation; saving storage space as well as processing time. Further, with the added timing precision we have found these tools to be well suited for one task in particular, detecting low-rate periodic anomalies.

Low-rate periodic anomalies occur in Internet traffic for a number of reasons, as will be discussed in Chapter 4. Low-rate anomalies are characterized as involving relatively few packets per second, as compared to the Internet traffic they are aggregated with. Due to the low-rate nature of the anomalies, detection requires examining long segments of data in order to gather enough evidence to be certain of an anomaly. The formulations presented here are computed using a fixed window size that is shifted over the long segment of data allowing efficient computation regardless of the length of the data being analyzed. Conversely, time series based signal processing methods, such as the fast Fourier transform, must be computed using a window size equal to the length of the data segment being analyzed and thus do not scale as efficiently as our formulations with increased trace durations.

Using the renewal theory formulations, we designed a system to detect low-rate periodic network anomalies, called inter-arrival based anomaly detection, or IA^{2}D. IA^{2}D detects differences between an estimate of the renewal density, computed from the Internet traffic measurements, and an "anomaly-free" distribution that we call the "smooth approximation". The smooth approximation is computed on the same measured data as the renewal density estimate, but by using certain characteristics of the anomaly we are able to filter out its presence in the smooth approximation.

The benefits of our IA^{2}D system include: 1) it can detect very low-rate periodic anomalies in aggregate Internet traffic measurements by only analyzing the timing of the events, 2) it operates without having to train on known anomaly-free measurements, and 3) our detection system can distinguish between multiple periodic events, which is not possible in other methods.

In addition to the renewal theory formulations presented here to analyze Internet traffic, renewal theory also provides us with another concept, called thinning, which is a method for subsampling a point process.The theory of thinning contains some particularly useful results for Internet analysis. One result gives the impact of thinning on the renewal density. Another outlines how certain renewal processes, when thinning is applied, converge towards the more ideal Poisson process. When we apply the second result to subsampling of Internet traffic measurements it allows us to analyze Internet measurements as if traffic were an ideal Poisson process, and use many of the existing results from renewal theory. In particular, since IA^{2}D is based on renewal theory we can use an ideal Poisson process to derive theoretical values for many of the system parameters, which can then be used, with modification, for real Internet traffic.

The first half of this chapter, Section 3.2 presents the necessary background in renewal theory for its use in analyzing Internet traffic. In this half of the chapter two formulations, the renewal function and renewal density, are described which are useful for analyzing long term behavior of Internet traffic. These formulations are designed for renewal processes, which are in general infinite length, we discuss the methods we use to estimate them for Internet traffic measurements. Finally, in the first half of this chapter, we discuss the theory of thinning, or subsampling, a renewal process and the impact of thinning on the renewal function and renewal density.

With the necessary background the second half of this chapter describes the design of our IA^{2}D detection system in Section 3.3. In this discussion we describe how we split the renewal density into sub-densities, which allows us to detect and distinguish multiple periodic anomalies. Then we discuss the operation of our statistical test, Pearson's Chi-Square test, that we use for detection, including how we construct the smooth approximation used for comparison in the test. Our detection system operates using a jumping window approach that allows the system to output detection decisions at fixed intervals specified by the user. Finally, we describe how to select various system parameters based on the assumption that the Internet traffic is modeled by a Poisson process. In Chapter 4 we discuss how to modify these system parameters when using actual Internet traffic because of the deviation of Internet traffic from the Poisson process assumption.

3.2 Background

3.2.1 Renewal Theory

We start by introducing some basic concepts in renewal theory as they relate to Internet traffic analysis. Most of this introduction comes from [55], which is a good source that describes how renewal theory can be applied to computer network and Internet traffic analysis.

Let $M, M_1, M_2, ..., M_n$ be independent and identically distributed non-negative random variables (M is a generic random variable of the sequence), reflecting the inter-arrival time between events. Note that high-rate measurement systems (i.e., no subsampling) record timestamps for each packet arrival, therefore, to generate a point process we replace the timestamp for each packet arrival with the time elapsed since the previous packet, e.g. $M_1 = M(2) - M(1), M_2 = M(3) - M(2), ..., M_n = M(n+1) - M(n)$.

Define the partial sums, $S_0 = 0, S_n = S_{n-1} + M_n = M_1 + M_2 + ... + M_n$, so that S_n, reflects the time between the starting time and n-th event. The renewal process, $N(t), t \geq 0$, is then defined to be the counting measure given by [55]:

$$N(t) = \sum_{n=0}^{\infty} I(S_n \leq t) = n : \; S_n \leq t \tag{3.1}$$

Here I is the indicator function, and thus $N(t)$ is a continuous time function that conveys the number of events that occur before time t. By studying the distributions of the partial sums we can learn the long range behavior of a point process.

For example, consider the problem of determining whether a point process consists of completely random events, i.e., a renewal process, or if the point process is a combination of a deterministic sequence, i.e. low-rate periodic events with period P, mixed with random events, as shown in Figure 3.1. If $P \gg \lambda$, the periodic events do not occur consecutively; rather they arrive with a random number of renewals between each periodic pair. The periodic events correspond to a subset of the inter-arrival times, i.e., $M_{i(k)}$ where $i = \{i(1), i(2), ..., i(k)\}$ is an ordered subset of $\{1, 2, ..., n\}$. In Figure 3.1, $i = \{1, 4\}$ and the periodic events correspond to M_1 and M_4. Then in terms of partial sums the inter-arrivals between periodic events gives $S_{i(j)} - S_{i(j-1)} = P \; \forall j \in \overline{[2, k]}$, e.g., $S_4 - S_1 = \{M_4 + M_3 + M_2 + M_1\} - \{M_1\} = \{M_4 + M_3 + M_2\} = P$ in the figure.

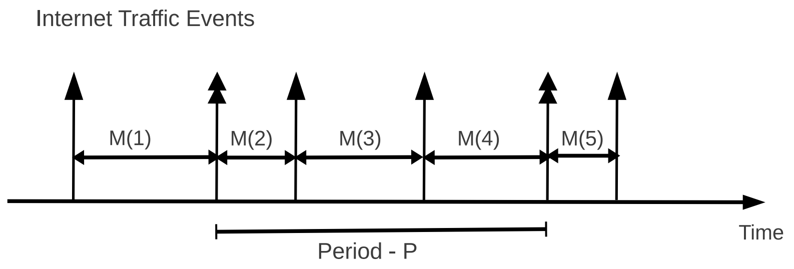

Figure 3.1: Point Process with Periodic Deterministic Sequence

3.2.1.1 Thinning a Renewal Process

One interesting area of Renewal Theory is 'thinning' of a Renewal Process, which is a method to subsample the data in a Renewal Process. The most common implementation of thinning, proposed by Rényi [62], is uniform probabilistic thinning where a measurement is retained with probability p and removed with probability $1-p$. Two main results from [62] that are useful in our work are:

1. when thinning a Poisson process with scale parameter λ, the resulting thinned process is also Poisson with scale parameter λ/p

2. for a general renewal process with intensity λ, the process obtained by thinning by a factor $1/p$ converges to a Poisson process with intensity λ/p as $p \longmapsto \infty$

The first result will be used in Section 3.2.3.3 when we look at the impact of thinning on the renewal density of a Poisson process. The second result is important because actual Internet traffic measurements do not follow a Poisson process, however, after applying subsampling (the measurement equivalent of thinning) Internet traffic measurements converge towards a Poisson process. The benefit of this results is that we can analyze the performance of our detection system using a Poisson process (Section 4.3.1), and determine some of the system parameters based on this analysis.

While Rényi only considered uniform probabilistic thinning, more general methods for thinning were examined in [21], including thinning using a thinning probability distribution. Thinning using a probability distribution is very similar to our idea of using a probability distribution for subsampling developed in Section 2.3. In fact, thinning is the equivalent operation to subsampling, however, the term subsampling is used to indicate when we are applying it to events or measurements whereas thinning is applied to renewal processes. One difference between our subsampling implementation and thinning is that when thinning is applied some measurements are discarded, and the information about how many measurements were discarded is not recorded. When working with renewal processes, which are infinitely long, discarding this information is valid because the resulting process after thinning remains infinitely long and computing (not estimating) the renewal density does not require information about the discarded measurements. In contrast, working with actual measurements (finite amount of data) makes it necessary to use information about discarded measurements when we estimate the renewal density, as described in Section 3.2.3. Therefore, in our work, we retain the number of measurements discarded.

For our theoretical analysis in Section 4.3.1, we limit the use of thinning theory to Rényi's thinning implementation because the two results mentioned above only work in a limited number of thinning probability distributions featured in [21]. However, in the future it could be possible to use the results from [21] to analyze the performance of our detection system for a more general class of subsampling (thinning) probability distributions. This would allow more general subsampling methods using the optimization techniques from in Section 2.3 to be analyzed in a similar manner as to what was done in Section 4.3.1.

3.2.2 Renewal Density

Let F_n be the cumulative distribution function (cdf) of the partial sum with n variables, i.e., F_n is the cdf of S_n. The renewal function, $R(t)$ is defined to be the expected number of renewals in a given time t, i.e., $R(t) = E[N(t)]$ which was shown in [55] to be:

$$R(t) = \sum_{n=1}^{\infty} F_n(t) \tag{3.2}$$

In other words, $R(t)$ is the sum of the cumulative distribution functions of all partial sums of inter-arrival times.

The renewal density, is similar to the renewal function, but instead indicates the probability that a renewal occurs at a given time t. Let f_n be the probability distribution function (pdf) of the partial sum with n variables, i.e., f_n is the pdf of S_n. The renewal density, $r(t)$, as shown in [55], is the sum of the pdfs of all partial sums of inter-arrival times, given by:

$$r(t) = \sum_{n=1}^{\infty} f_n(t) \tag{3.3}$$

For our periodic example above we could expect that the renewal density would have spikes at multiples of the fundamental period, i.e., $t = kP$ for some integer k. The size of the peak is a function of the ratio of periodic traffic rate and the regular Internet traffic rate, and is discussed in more detail in Section 3.3.2.

The renewal function and density are related by:

$$\begin{aligned} f_n(t) &= \frac{\delta F_n(t)}{\delta t} \\ r(t) &= \frac{\delta R(t)}{\delta t} \end{aligned} \tag{3.4}$$

For renewal processes the renewal function and renewal density are completely characterized by the cdf or pdf of the random variable M. This occurs because, as shown in [55], the distribution of F_n is given by the n-fold convolution of the cdf of M, thus:

$$F_n(M) = F^{n*}(M) = \underbrace{F_M * F_M * ... * F_M}_{n\ times} \tag{3.5}$$

and similarly for the pdf $f_n(M)$. Therefore, knowing the cdf or pdf of M it is possible to compute the densities of all partial sums and computer either $R(t)$ or $r(t)$.

For a Poisson process, with mean inter-arrival time λ, the renewal function and renewal density are given by the simple expressions [55]:

$$\begin{aligned} R(t) &= \frac{t}{\lambda} \\ r(t) &= \frac{1}{\lambda} \end{aligned} \tag{3.6}$$

Thus the renewal function is a linear function and the renewal density is a constant equal to the inverse of the mean inter-arrival time of the process. The reason $r(t)$ is flat for a Poisson process can be seen in Figure 3.2, which shows the pdfs of the first few inter-arrival orders that are generated with a Poisson process. As the inter-arrival order increases the distribution shifts towards longer inter-arrival times, and the distribution becomes smoother. The shift causes the significant mass in the distributions not to overlap. The sum of all the shifted and smoothed distributions creates a flat renewal density show as the bottom distribution in Figure 3.2.

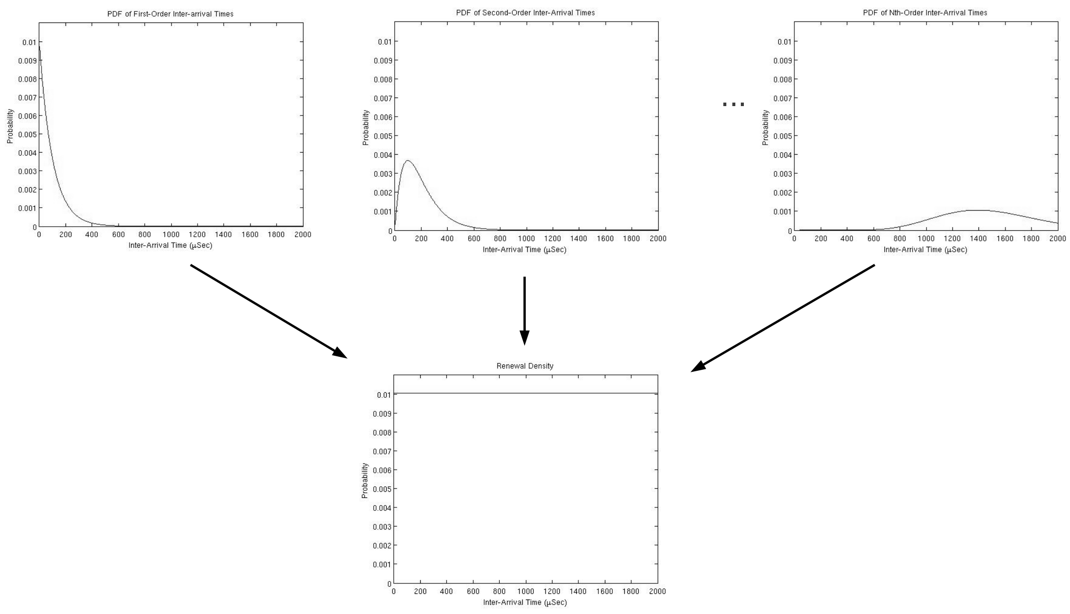

Figure 3.2: Generating renewal density for Poisson process

3.2.3 Estimating the Renewal Density

If f_M is unknown then most methods to estimate the renewal density rely on generating multiple instances of the point process to compute estimates of f_n and $r(t)$, denoted $\hat{f}_n$ and $\hat{r}(t)$ where we use $\hat{\cdot}$ to denote empirical estimates [29]. However, for practical signals, e.g., packet arrivals, it is not feasible to generate multiple realizations of the point process. Instead, given empirical measurements we generate time averages using observations from a single set of Internet traffic measurements, i.e., the renewal density estimate represents the average of multiple observations from different periods of time. This is accomplished by using multiple observations of the same set of measurements where each observation reflects a different starting point in the measured signal, a technique derived in [51].

The method from [51] operates by starting with the first event in the signal, M_0, and computing the sequence of partial sums up to the desired order n, i.e., $\{S_1(M_0), S_2(M_0), ..., S_n(M_0)\}$. Where the partial sums are given by:

$$\begin{aligned} S_1(M_0) &= M_1 \\ S_2(M_0) &= M_1 + M_2 \\ S_n(M_0) &= S_{n-1}(M_0) + M_n \end{aligned} \tag{3.7}$$

Moving to the next event in the signal, M_1, the process is repeated to generate the next sequence of partial sums, $\{S_1(M_1), S_2(M_1), ..., S_n(M_1)\}$.

The estimate $\hat{f}_n$ is computed non-parametrically using histograms that tabulate the data from each partial sum observation, $S_n(M_j)$, $\forall j$, for the entire set of observations. Each partial sum of order 1 to n is used in the estimate of the pdf at the respective order, i.e., $S_i(M_j)$ contributes to $\hat{f}_i$ only. Finally, $\hat{r}(t)$ is computed as:

$$\hat{r}(t) = \sum_{i=1}^{N} \hat{f}_i(t) \tag{3.8}$$

Figure 3.3 shows the operation of our renewal density estimation technique for a set of measurements. The operation is similar to that in Figure 3.2 except that we have to compute the partial sums from different observations (starting points) of the same set of measurements (shown in the top half of the figure) and estimate the pdfs, $\hat{f}_n$, using histograms. Instead, in Figure 3.2 the partial sums are computed using different realizations of the renewal process, and the pdfs are the actual distributions f_n.

When subsampling is applied to the measurement the estimation of the renewal density must take into account both components of the vector measurement signal,

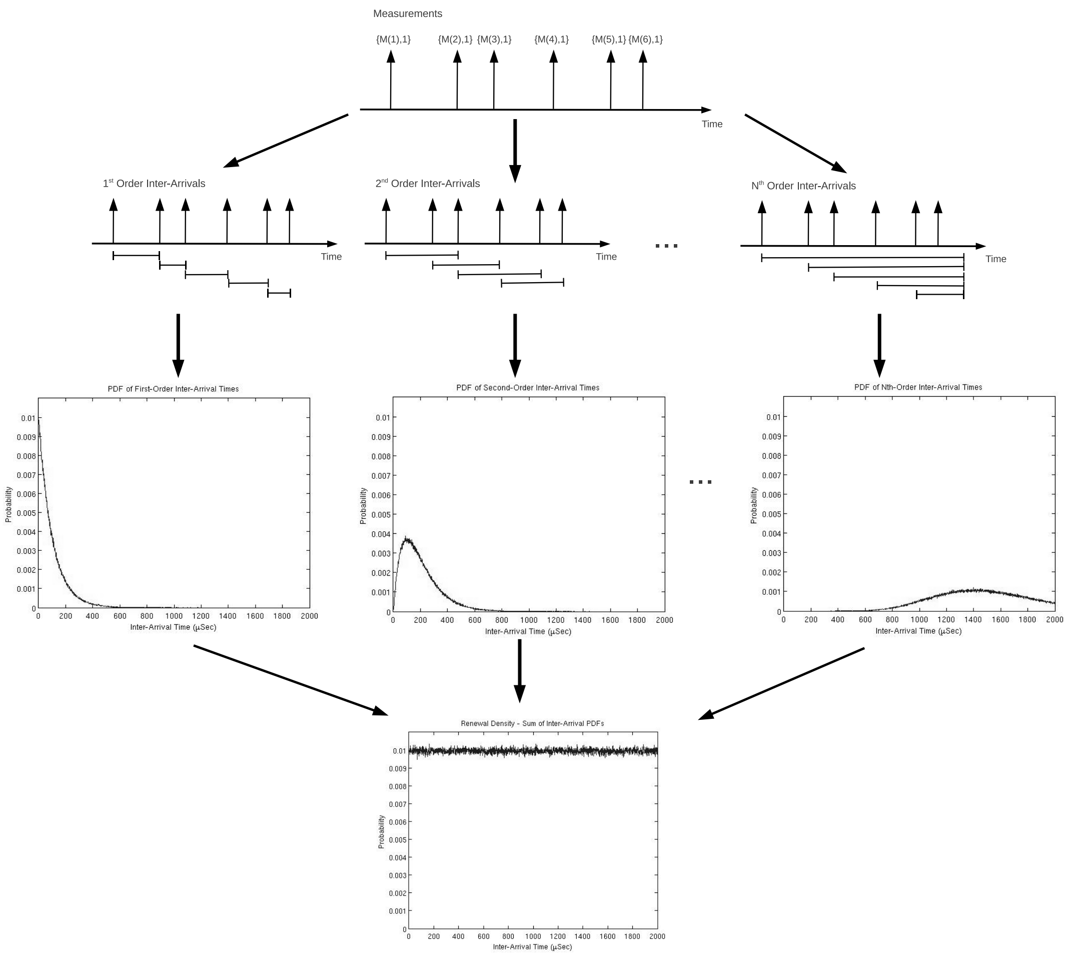
Measurements
{M(1),1}
{M(2),1}
{M(3),1}
{M(4),1}
{M(5),1}
{M(6),1}
Time
1st Order Inter-Arrivals
2nd Order Inter-Arrivals
Nth Order Inter-Arrivals
PDF of First-Order Inter-Arrival Times
PDF of Second-Order Inter-Arrival Times
PDF of Nth-Order Inter-Arrival Times
Probability
Inter-Arrival Time (μSec)
Renewal Density - Sum of Inter-Arrival PDFs

Figure 3.3: Generating renewal density estimate from measurements

i.e., the timestamps, $M(k)$, and the number of packets in that measurement, $C(k)$. Let $X(k)$ be the current measurement with components $M(k)$ and $C(k)$. The partial sum sequence for $X(k)$ is constructed as follows:

$$\begin{aligned} S_0(M(k)) &= 0 \\ S_{n(1)}(M(k)) &= M(k-n(1)) \\ S_{n(i)}(M(k)) &= S_{n(i-1)}(M(k)) + M(k+i) = ... \\ &= M(k+1) + M(k+2) + ... + M(k+n(i)) \end{aligned} \tag{3.9}$$

Where $n(i)$ is computed to be the sum of the $C(k)$'s:

$$n(i) = C(k+1) + C(k+2)... + C(k+i) \tag{3.10}$$

Note that in the sum of $n(i)$ the range is from $k+1$ to $k+i$. Thus, $S_{n(i)}(M(k))$ reflects the time between the current measurement and $n(i)$-th previous event.

With this modification, estimating the renewal density using the technique from [51] takes into account that for each input measurement the partial sums generated only contribute to the estimates of a subset of the density functions $\hat{f}_{n(i)}$.

Using the estimation method from [51] the observations are not independent, as they contain a majority of the same inter-arrival times. The estimation method introduced by Frees in [29] generates additional observations of the point process by rearranging the order of the original inter-arrival times, where each observation corresponds to a different permutation of the indices n of the sequence t_n. The method from [29] works well for renewal processes where each inter-arrival time is independent and identically distributed. However, the Internet traffic measurements we consider are not guaranteed to be independent or identically distributed. For example, when a packet is queued at a router it is output as soon as the router has finished processing the previous packet. In this example the inter-arrival time of the queued packet depends on the time taken to process the previous packet and therefore is not independent. Since Internet traffic measurements are not ideal renewal processes the method from [29] cannot be used to estimate the renewal density of Internet traffic.

The fact that Internet traffic measurements are not a renewal process does not impact our use of the renewal density estimate for detection. We use the renewal density estimate as a means to examine the long range timing behavior of a point process and to detect anomalies in the timing behavior. Computing a renewal density estimate using the technique from [51] does not require that the measurements come from a renewal process. Further, we use properties of the renewal density that are specific for renewal processes (such as generating f_n from f_1 via convolution) only in our analysis section where we approximate Internet traffic by a Poisson process, and in Chapter 4 we discuss modifications to this analysis specific for Internet traffic analysis.

3.2.3.1 Histogram as Density Estimator

We now discuss the estimation of the renewal density and the inter-arrival probability density functions when a histogram approach is used. In [28], the authors discuss the errors inherent to estimating a probability density using a histogram. The authors use a binomial random variable to model the number of events falling in the j-th histogram bin interval, e.g., $\{t_0 + j \cdot h, t_0 + (j+1) \cdot h\}$, where t_0 is a reference point and h is the bin width in seconds. Letting N_j be the number of events falling in the j-th bin, then the value of the histogram, $\hat{f}(t)$, at the point $t \in \{t_0 + jh, t_0 + (j+1)h\}$ is given by $\hat{f}(t) = N_j/mh$ where m is the total number of events, thereby normalizing $\hat{f}(t)$ so that the area under the histogram is unity.

Let $f_h(t)$ be the integral of the actual probability density (the one being estimated) over the bin interval containing the point t, i.e.,

$$f_h(t) = \int_{t+jh}^{t+(j+1)\cdot h} f(t)dt \; . \tag{3.11}$$

Let $f_h^i(t)$ denote the integral, over the same bin interval as above, for the pdf of the

i-th order partial sum sequence. Further, let $r_h(t)$ be the integral, over the same bin interval as above, for the renewal density, then

$$r_h(t) = \int_{t+jh}^{t+(j+1)\cdot h} r(t)dt \tag{3.12}$$

or in terms of $f_h(t)$:

$$r_h(t) = \sum_{i=1}^{\infty} f_h^i(t) \tag{3.13}$$

Here we assume that the bin interval, h, is the same for all $f_h^i(t)$ otherwise the summation would not be possible.

Using the definition of $f_h(t)$ from (3.11), the number of events in the j-th histogram bin, N_j, is a binomial random variable with number of trials m and success probability $f_h(t)$ for $t \in \{j\cdot h, (j+1)\cdot h\}$. Finally, the expected value and variance of the histogram, $\hat{f}(j)$ are given by [28]:

$$\begin{gathered} E[\hat{f}(j)] = f_h(t) \\ Var[\hat{f}(j)] = \frac{1}{mh} \cdot f_h(t)(1 - h \cdot f_h(t)) \\ t \in \{j \cdot h, (j+1) \cdot h\} \end{gathered} \tag{3.14}$$

We now consider how to compute an estimate of the renewal density, $\hat{r}(j)$, which is our novel adaptation of the probability density estimator work in [28]. This involves the sum of multiple histogram estimates, so that the expected value of the renewal density estimate is given by:

$$\begin{gathered}
E[\hat{r}(j)] = E[\sum_{i=1}^{\infty} \hat{f}_i(t)] = \sum_{i=1}^{\infty} E[\hat{f}_i(t)] \\
E[\hat{r}(j)] = \sum_{i=1}^{\infty} f_h^i(t) = r_h(t) \\
t \in \{j \cdot h, (j+1) \cdot h\}
\end{gathered} \tag{3.15}$$

In other words the expected value of the renewal density estimate is the integral of the actual renewal density over the bin interval, i.e., $r_h(t)$. Thus, using the histogram approach the renewal density estimate is an unbiased estimator of the actual renewal density.

Finding an expression for the variance of the renewal density estimate is not as straight-forward as was finding the expected value. The variance of the estimate is given by:

$$Var[\hat{r}(j)] = Var[\sum_{i=1}^{\infty} \hat{f}_i(j)] \ , \tag{3.16}$$

where, even though the renewals in the process are independent, the probability density function of the partial sum sequence, say S_k, depends on the pdf of the partial sum at the previous order, S_{k-1}, which in turn depends on all previous lower orders. This also applies to the histogram based estimates of the partial sum pdfs, i.e., the $\hat{f}_i(j)$. Thus, the variance of the estimate can not be readily simplified beyond the above expression.

Alternatively, rather than find the variance of the sum of multiple histogram estimates we can approximate this by treating the sum of the histograms as a single histogram, which is the estimate of our renewal density, $\hat{r}(j)$. In other words instead of computing the variance using (3.16) we find the variance by:

$$Var[\hat{r}(j)] = Var[r_h(t)] \;\; t \in \{j \cdot h, (j+1) \cdot h\} \ . \tag{3.17}$$

Then applying Equation (3.14) we can express the variance of the renewal density estimate $\hat{r}(j)$ as:

$$Var[\hat{r}(j)] = \frac{1}{mh} \cdot r_h(t)(1 - h \cdot r_h(t)) \ . \tag{3.18}$$

This approximation is suitable given that for a specific bin interval, $\{jh, (j+1)h\}$, only a small number of histogram estimates, $\hat{f}_i(j)$, contribute a non-negligible probability density. This approximation is based on the fact that as the order i increases the probability mass of $\hat{f}_i(j)$ is shifted towards longer inter-arrival times as was shown in Figure 3.3. Because only a few $\hat{f}_i(j)$ are non-negligible we can treat the $\hat{f}_i(j)$ as independent over the small bin intervals. Therefore the variance of the sum, $Var[\sum \hat{f}_i(j)]$, is approximated by the sum of the variances, $\sum Var[\hat{f}_i(j)]$.

In Section 3.3, we introduce the Pearson Chi-Square test [33], which is the statistical test we use for anomaly detection. The Pearson Chi-Square test requires the number of occurrences and not probability of occurrence, which is given by $\hat{f}_i(j)$. Therefore the histogram estimates used in Section 3.3 are not normalized by the total number of measurements, m, and we denote them by, $\tilde{f}_i(j)$. Similarly the renewal density estimate computed using the $\tilde{f}_i(t)$ is denoted, $\tilde{r}(t)$. This modifies the expected value of the renewal density estimate from the expression in (3.15) to the following:

$$\begin{gathered}E[\tilde{r}(j)] = E[\sum_{i=1}^{\infty} m_i \cdot \tilde{f}_i(t)] = \sum_{i=1}^{\infty} E[m_i \cdot \tilde{f}_i(t)] \\ E[\tilde{r}(j)] = \sum_{i=1}^{\infty} m \cdot f_h^i(t) = m \cdot r_h(t) \\ t \in \{j \cdot h, (j+1) \cdot h\} \ ,\end{gathered} \tag{3.19}$$

and the variance of the renewal density estimate, using the same approximation that was used in (3.18), is given by:

$$\begin{gathered}Var[\tilde{r}(j)] = \frac{m}{h} \cdot r_h(t)(1 - h \cdot r_h(t)) \\ t \in \{j \cdot h, (j+1) \cdot h\} \ ,\end{gathered} \tag{3.20}$$

Note that the number of events at a particular order, m_i, is replaced with m, which is the average number of events. This approximation is made because in practice typical values of m_i are very large, so small differences between a particular m_i and the average m are negligible.

3.2.3.2 Convergence of Renewal Density Estimate

We now consider the convergence of the histogram to density being estimated. This topic is covered in detail in [28] where the author considers the mean square deviation between the histogram and the density being estimated as a function of the size of the histogram bins, h. Note that [28] only considers estimation of a single density function, and not the sum of multiple density function estimations that is required for computing the renewal density estimate. Some comments on how convergence of the renewal density estimate will be discussed at the end of this section. The optimal size of h naturally depends on the shape of the density being estimated and the number of samples used to generate the histogram. For example, if the number of samples, m, is small and we select h to be narrow then most histogram bins will contain only a few samples and the histogram will not be a good indication of the underlying probability distribution. Similarly, if the curve of the distribution is flat, e.g., a uniform distribution, then we do not need many histogram bins in order to achieve an accurate representation of the distribution; in fact for a uniform distribution only one histogram bin would suffice. On the other hand, for complex distributions, such as an exponential or Laplacian distribution, many more histogram bins are required to accurately model the complex curvature of these distributions.

As a guideline for selecting, h, in [28] the authors propose the following "rule": *Choose the cell width as twice the interquartile range (IQR) of the data divided by the cube root of the sample size, i.e.,*

$$h_{min} \geq \frac{2 \cdot IQR}{m^{1/3}} \tag{3.21}$$

This rule captures both conditions mentioned above. The interquartile range, IQR, measures the dispersion of the distribution, which gives an indication of the smoothness of the distribution. While dividing by $m^{1/3}$ keeps the width of h wider when m is small, and h can be narrow as m increases.

When selecting the histogram bin width for our detection system, which we discuss in Section 3.3.5, we follow this rule but provide some modifications since we know the distributions that we are trying to estimate using the histograms. When selecting the initial system parameters we make the assumption that Internet traffic can be modeled as a Poisson process. Using the Poisson process assumption, the distribution of inter-arrival times at order n follows an Erlang, or Gamma distribution with scale parameter λ and shape parameter equal to the order of the inter-arrival, i.e., for $f_n(t)$ the shape parameter is n. For large n we can use the Normal approximation to the Gamma distribution, i.e., $f_n(t) \sim N(n \cdot \lambda, n \cdot \lambda^2)$. The interquartile range for the Normal distribution is simply $IQR = 1.34\sigma$, which given our approximation would be $IQR = 1.34 \cdot \sqrt{n \cdot \lambda^2}$. Using this approximation we modify (3.21) to derive the following expression for the minimum size of h:

$$h_{min} \geq \frac{2.68 \cdot \sqrt{n \cdot \lambda^2}}{m^{1/3}} \tag{3.22}$$

A more convenient expression for h_{min} is to write it as a function of time. We can accomplish this by replacing m with the average number of inter-arrival measurements per second, where $m = t/\lambda$, and rewrite (3.22) as:

$$h_{min} \geq \frac{2.68 \cdot \sqrt{n} \cdot \lambda^{4/3}}{t^{1/3}} \tag{3.23}$$

Using this version of the equation it is easier to determine if the histogram will sufficiently converge in the amount of time that our system uses to detect an anomaly (discussed in Chapter 4).

As mentioned previously the convergence results from [28] were derived for the estimation of a single density function, i.e., $f_n(t)$ for some n. Since, in order to estimate the renewal density, we have to estimate multiple density functions the result for h_{min} in (3.23) does not guarantee that the entire renewal density will converge. Instead we use the approximation, described in Section 3.2.3.1, that inside a single sub-density only a few of the inter-arrival density function estimates, i.e., only a few $\tilde{f}_n(t)$, are non-zero. Therefore, we can use the convergence criterion, given by (3.23), to verify that the non-zero density function estimates being used in the sub-density where we are performing detection have sufficiently converged. For the non-zero density functions in a particular sub-density, i.e., the n such that $\tilde{f}_n(t) > 0$, it is possible check that the value of h_{min} is satisfied because each sub-density has its own time-to-detection (described in Section 3.3.2). Thus, we can use the time-to-detection as the value of t and the values of n corresponding to the non-zero density functions to check that h_{min} is satisfied using (3.23).

3.2.3.3 Effect of Thinning on Renewal Density

When thinning is applied to a renewal process the renewal density is altered. For a Poisson process with scale parameter λ, using the result from Rényi [62] mentioned in Section 3.2.1.1, when thinned with uniform probability, $1/\mu$, i.e., thinning rate is μ, the resulting process remains Poisson with scale parameter $\mu \cdot \lambda$. The renewal density of a Poisson process was given in [55] to be $r(t) = 1/\lambda$. Combining the results we see that the renewal density of the process resulting after thinning a Poisson process with scale parameter λ by a factor of μ is simply:

$$r_\mu(t) = \frac{1}{\mu \cdot \lambda} \tag{3.24}$$

Using the second result from Rényi [62], that a general renewal process with intensity λ when thinned by a factor of μ converges to a Poisson process with scale parameter $\lambda' = \mu \cdot \lambda$ as $\mu \longmapsto \infty$, we can see that by thinning a more general renewal process its renewal density converges to (3.24) as $\mu \longmapsto \infty$. This feature of renewal theory will be used when analyzing and implementing the detection system for real Internet traffic in Section 4.3.1. Real Internet traffic does not behave like a Poisson process, and because of this the renewal density estimated for Internet traffic is not a constant function as defined in (3.6). However, by applying thinning to the measurements the renewal density estimated for Internet traffic converges to the flat distribution as indicated by (3.6). The trade-off being that, when thinning is applied, it takes longer for the histograms used to compute the renewal density estimate to converge. This is clear from (3.23), where the effective scale parameter, λ', after thinning becomes larger and requires a longer time, t, before the histogram will converge.

3.3 Detection of Periodic Events

As discussed in Section 3.2.2 the renewal function and density are two important tools used to characterize the long term behavior of point processes. Using techniques to estimate these functions for empirical measurements of Internet traffic allows us to detect anomalous behavior in the Internet measurements, even if the behavior occurs infrequently. One particular task that the renewal density is well suited for is detecting low-rate, periodic events interspersed in regular Internet traffic. Multiple causes of periodic events in Internet traffic will be discussed in Chapter 4 and include malicious activity such as DoS attacks, as well as benign but undesirable network behavior such as bottleneck links.

There are three main classes of methods that have been applied to detect low-rate periodic events: frequency domain, rate-based, and inter-arrival timing based. Of the three, frequency domain methods are the most common and work by transforming the signal into the frequency domain then searching for frequencies containing significant energy, which can signal the presence of an anomaly. Such methods have been considered before, see [34], to detect period packet arrivals in Internet traffic. In [52] we showed that our inter-arrival timing based method, called periodic detection with multiple measurements (PDMM), outperformed the best frequency domain method available at that time [34] when detecting low-rate anomalies. PDMM is a precursor to the detection method described in this chapter.

More recently Thatte, *et al.* derived a rate-based method, called the bivariate parametric detection mechanism (bPDM) [75]. Rather than using a point process representation, where observed arrival times are used, bPDM first applies a coarse uniform time sampling to generate a signal that records the number of events (packet arrivals) per unit time. Using this signal bPDM derives statistics to determine if the events are random or random plus a constant signal; signifying the presence of a constant rate anomaly (an example of which would be a periodic anomaly). Similar to our previous results, in [75] bPDM was shown to be superior to frequency domain based methods.

In this section we propose, as an alternative to frequency domain and rate-based methods an inter-arrival time based method, which we call inter-arrival based anomaly detection, IA^{2}D. Our detection system works by applying Pearson's Chi-Square test on the renewal density [33]. The detection method makes use of the techniques shown in Section 3.2.3 to estimate the renewal density. Since the periodic events only affect a portion of the estimated renewal density it makes sense to divide the renewal density into smaller segments, which we call sub-densities, and perform detection on the individual sub-densities; this process is described in Section 3.3.1. In order to use Pearson's Chi-Square test, described in Section 3.3.2, we require a second renewal density that is anomaly-free, which we term the "smooth approximation". In Section 3.3.3, methods to estimate this anomaly-free distribution from the empirical measurements, even in the presence of an anomaly, are discussed. Finally, the operation of the detection method depends on the period itself, i.e., it takes longer to detect anomalous events with longer periods. Therefore, we use jumping windows with varying lengths, discussed in Section 3.3.4, that are used to estimate the renewal density in the different sub-densities and allow our system to output detection decisions at regular intervals regardless of the period of the anomaly. By performing detection independently on individual sub-densities, IA^{2}D can detect and *distinguish* between multiple anomalies, something bPDM cannot do.

Finally, while not considered in this thesis, framing IA^{2}D in terms of renewal theory makes this method readily applicable to many scientific signals of interest, where anomalies can be important for diagnosing disease in biological signals or may be evidence of potential malicious activity in financial data, e.g. stock market data, [60].

3.3.1 Sub-Densities

One difficulty with using the renewal density to detect low-rate periodic events is that it requires estimating the renewal density, $r(t)$, up to and beyond the period of the event, e.g., $t > P$. Without prior knowledge about the period of the events, which is assumed unknown, this requires that detection be performed for all potential periods. Further, the periodic anomaly only affects a portion of the renewal density in the immediate area around $t = P$, and the remainder of the renewal density is unaffected by the anomaly. The unaffected portion of the renewal density decreases the performance of our detection method because the test we use for detection, Pearson's Chi-Square test, compares the renewal density estimate to an anomaly free distribution, and computes a score based on how similar the two distributions are. Pearson's test compares the anomaly free and anomalous distributions over the entire range, i.e., $0 \leq t \leq P$, and generates a score based on the difference between the distributions, however, this score is normalized by the width of the range compared. Therefore, the normalized score would be much smaller when the entire range is considered versus when a smaller range, i.e., $P - \delta \leq t \leq P + \delta$, is used because the smaller range would produce the same difference score (before normalization) as the entire range. Because of this we designed our detection system such that the renewal density is divided into smaller blocks, which we call sub-densities, e.g., $\tilde{r}_s(j)$ $s \in [1, NSub]$ where $NSub$ is the number of sub-densities Detection is then performed independently in each sub-density and because range used in the normalization is smaller we improve detection performance by using sub-densities.

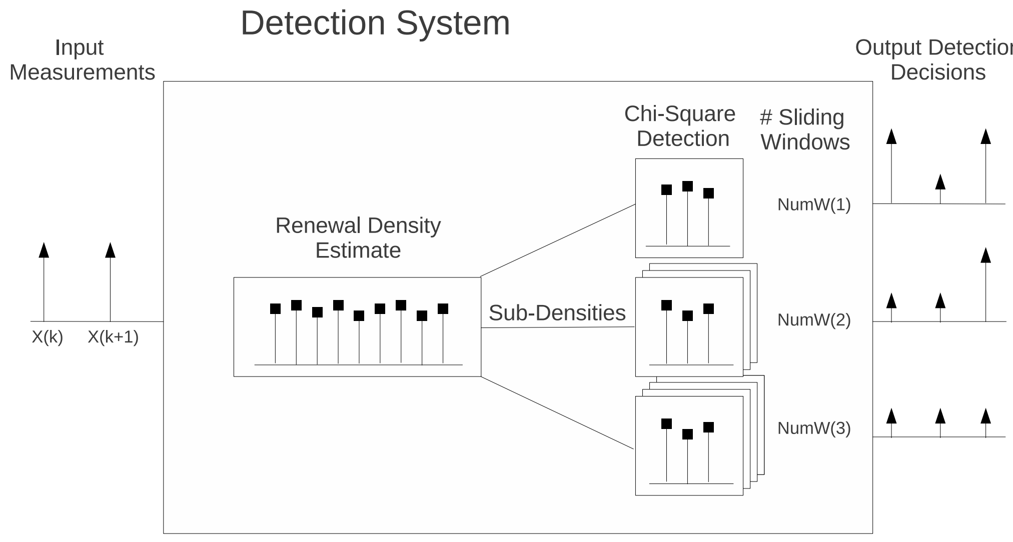

Figure 3.4: Design of Detection System

The detection system is designed as shown in Figure 3.4. Where the input to the system are the measurements obtained from the measurement system, and the multiple outputs correspond to detection decisions from the respective sub-densities. Performing detection on sub-densities independently presents an interesting system design challenge. The number of events that must be observed in order to be confident, statistically speaking, and decide whether or not the traffic contains an anomaly depends on the period of the event. Thus, detecting higher-rate events (with smaller periods P), which arrive at the system more frequently, requires less time than does detection of lower-rate events. Therefore, sub-densities corresponding to shorter periods can potentially report detection decisions more frequently. In our system, however, it is desirable to have *all* sub-densities report detection decisions at the same time interval. The solution to this, which is explained in Section 3.3.4, is to observe events using jumping windows of varying length. Thus, all sub-densities can report detection decisions at the same time.

Processing inside the detection system proceeds as follows. The input measurements are processed individually. For each input measurement the estimate of the renewal density $\tilde{r}(j)$ is updated using the technique discussed in Section 3.2.3. If the time since the last detection decision is greater than the output time interval then detection is performed in all sub-densities. Detection is performed using Pearson's Chi-Square test, described in Section 3.3.2, and the result of the test, whether or not a periodic event was detected, is then output from the system. After the result is output the oldest observation window, described in Section 3.3.4, is discarded for all sub-densities and the system resumes processing input measurements until the next output time interval occurs.

3.3.2 Pearson's Chi-Square Test

For detection we use Pearson's Chi-Square test [33] in order to determine the goodness-of-fit between the renewal density estimate computed empirically, $\tilde{r}(j)$, and a density derived that is free of anomalies. The anomaly-free distribution, termed the "smooth approximation" and denoted $\tilde{s}(j)$, can intuitively be thought of as what would be the value of $\tilde{r}(j)$ at each j without the periodic traffic. Two methods to derive the "smooth approximation" are described in Section 3.3.3, using knowledge about the characteristics of the anomaly being detected.

After obtaining the two renewal density approximations the Pearson Chi-Square test is applied to the sub-densities, denoted $\tilde{r}_s(j)$ and $\tilde{s}_s(j)$. Let N_{bin} be the number of histogram bins in each sub-density. Then χ_i^2 is computed for the i-th sub-density, $\tilde{r}_{s(i)}(j)$, using the equation:

$$\chi_i^2 = \sum_{m=1}^{N_{bin}} \frac{(\tilde{r}_{s(i)}(m) - \tilde{s}_{s(i)}(m))^2}{\tilde{s}_{s(i)}(m)} \tag{3.25}$$

Next the Pearson statistic, p, is calculated. This is the value of the cdf of a chi-square distribution with N_{bin} degrees of freedom computed at the value χ_i^2. The null hypothesis, that the empirical renewal density estimate is statistically similar to the smooth approximation, is rejected if $p > 1 - P_{FA}$ where P_{FA} is the probability of false alarm and for our application is typically selected to be 0.01 or less [33].

When implementing the detection system in practice, evaluating the cdf of the χ^2 distribution in real time is too computationally expensive. Assuming P_{FA} and N_{bin} remain fixed, a more convenient solution is to pre-compute the value of χ_i^2 which is required to select the alternate hypothesis, i.e., that an anomaly is present. We call this value χ_D^2 (it is described later in this section and given by (3.34)). By pre-computing the value of χ_D^2 the selection of a hypothesis is done by comparing χ_i^2 to χ_D^2 instead of having to evaluate the cdf of a chi-square distribution.

To understand in more detail why the sum in (3.25) follows a χ^2 distribution and to evaluate the detection system performance we analyze the operation of the detector in the null and alternative hypothesis cases. Before we begin, however, we make a modification to our renewal process assumption that is essential to the following analysis.

Recall from Section 3.2.3.1, that the value of the histogram used in the renewal density estimate at bin j in sub-density i, i.e., $\tilde{r}_{s(i)}(j)$, is a binomial random variable with mean $m \cdot r_h(t)$ and variance $\frac{m}{h} \cdot r_h(t)(1 - h \cdot r_h(t))$ (for $t \in \{j \cdot h, (j+1) \cdot h\}$ see (3.19) and (3.20)). When the mean and variance were derived we assumed that the renewal process was based on a continuous distribution, i.e., the distribution of the renewals f_M was a continuous distribution (e.g., an exponential distribution for a Poisson process). However, in practice the measurements of packet arrival times are sampled at discrete time intervals. In other words, the value $M(k)$ from (2.1) is some multiple of the measurement sampling interval, T_s, which typically is in μseconds. We use $\cdot'$ to denote the discrete time version of a continuous time variable, e.g., t' is the discrete time sampled version of t. Further, let $\bar{\cdot}$ be the sample number, i.e., the multiple of the sampling interval, T_s, corresponding to the discrete time variable, e.g., $\bar{t} \cdot T_s = t'$. Written another way:

$$\bar{t} = t'/T_s \tag{3.26}$$

$\bar{t}$ is an integer valued number, as are all values denoted with $\bar{\cdot}$.

We use this notation because we can modify the expression $t \in \{j \cdot h, (j+1) \cdot h\}$ to be $\bar{t} \in \{j \cdot \overline{h}, (j+1) \cdot \overline{h}\}$. Then by selecting the value of $h = T_s$ the value of $\overline{h}$ becomes $\overline{h} = h/T_s = 1$. Thus when working with the sample numbers, the expressions for the mean and variance of $\tilde{r}_{s(i)}(j)$ (from (3.19) and (3.20)) become:

$$\begin{aligned} E[\tilde{r}_{s(i)}(j)] &= m \cdot r_h(\bar{t}) \\ Var[\tilde{r}_{s(i)}(j)] &= \frac{m}{1} \cdot r_h(\bar{t})(1 - 1 \cdot r_h(\bar{t})) \ . \end{aligned} \tag{3.27}$$

Finally, because the value of $r_h(\bar{t}) \ll 1$ we can approximate $Var[\tilde{r}_{s(i)}(j)]$ as:

$$Var[\tilde{r}_{s(i)}(j)] \approx m \cdot r_h(\bar{t}) = E[\tilde{r}_{s(i)}(j)] \tag{3.28}$$

Note that this assumption is required for our analysis in the following sections, in (3.30) and (3.31), and does not change any of the analyses that were conducted in Sections 3.2.3 or 3.2.3.1.

3.3.2.1 Null Hypothesis - No Anomaly

We begin our analysis of the Pearson chi-square detector by considering why in the null hypothesis scenario the chi-square score computed using (3.25) follows a centralized $\chi^2(1)$ distribution. In this and the next section we drop the index i on the sub-density, i.e., use s instead of $s(i)$, to simplify the expressions knowing that the analysis (and detection) is performed one sub-density at a time.

If we assume that the number of observations, m, is large then the binomial random variable, $\tilde{r}_s(j)$, can be approximated with a normally distributed random variable (having mean and variance given by (3.27)). According to [32], for a normally distributed random variable, X, with mean ω and variance σ^2, we have that Y defined as:

$$Y = \frac{(X - \omega)^2}{\sigma^2} \tag{3.29}$$

follows a $\chi^2(1)$ distribution, i.e., $Y \sim \chi^2(1)$ with $E[Y] = 1$ and $Var[Y] = 2$.

Therefore, if the empirical renewal density and smooth approximation estimates correspond to the same renewal density, i.e., no anomaly present, then according to (3.27):

$$\begin{aligned} X_j &= \tilde{r}_s(j) \\ \omega_j &= E[\tilde{r}_s(j)] = m \cdot r_h(\bar{t}) \\ \sigma_j^2 &= Var[\tilde{r}_s(j)] \approx m \cdot r_h(\bar{t}) = E[\tilde{r}_s(j)] \ , \end{aligned} \tag{3.30}$$

where we use the subscript j to denote that the mean and variance are taken over multiple realizations of the same histogram bin, i.e., multiple values of $\tilde{r}_s(j)$, observed over a period of time. If we assume that $\tilde{s}_s(j) \approx E[\tilde{r}_s(j)]$, which should be true since $\tilde{s}_s(j)$ is the smooth approximation of $\tilde{r}_s(j)$, then we can rewrite ω and σ^2 as:

$$\begin{aligned} \omega_j &= E[\tilde{r}_s(j)] \approx \tilde{s}_s(j) \\ \sigma_j^2 &= Var[\tilde{r}_s(j)] \approx E[\tilde{r}_s(j)] \approx \tilde{s}_s(j)\ . \end{aligned} \tag{3.31}$$

Finally, if we replace the values of X, ω and σ^2 in (3.29) with their values from (3.30) and (3.31), then we see that the expression in (3.29) is identical to a single term in the summation (3.25). Therefore, each term in the summation (3.25) is approximately $\chi^2(1)$ distributed, where $\chi^2(1)$ is a chi-square random variable with $E[\chi^2(1)] = 1$ and $Var[\chi^2(1)] = 2$.

The sum of N_{bin} $\chi^2(1)$ random variables is a $\chi^2(N_{bin})$ random variable with $E[\chi^2(N_{bin})] = N_{bin}$ and $Var[\chi^2(N_{bin})] = 2 \cdot N_{bin}$. Thus, the sum in (3.25) follows a χ^2 distribution with N_{bin} degrees of freedom.

3.3.2.2 Alternative Hypothesis - Anomaly

When an anomaly is present, let the estimates of $\tilde{r}_s(j)$ and $\tilde{s}_s(j)$ still be approximated by normal distributions, but each having a different expected value. This makes intuitive sense from an anomaly standpoint. If $\tilde{r}_s(j)$ corresponds to the value of the empirical renewal density containing the anomaly, then $\tilde{r}_s(j)$ is the combination of the anomaly-free distribution, $\tilde{s}_s(j)$, plus an additional component due to the anomaly. Denote γ_j to be the component of the renewal density estimate at $\tilde{r}_s(j)$ due to the anomaly:

$$\gamma_j = \tilde{r}_s(j) - \tilde{s}_s(j) \tag{3.32}$$

In the previous case because $\omega \approx \tilde{s}_s(j)$ and $\sigma^2 \approx \tilde{s}_s(j)$ applying (3.29) resulted in a $\chi^2(1)$ random variable. In the anomalous case, if the anomaly appears in histogram bin j, then for $X = \tilde{r}_s(j)$ we can no longer assume $\omega = \tilde{s}_s(j)$ and $\sigma^2 = \tilde{s}_s(j)$. Nor can we assume applying (3.29) results in Y_j following a non-central χ^2 distribution $\chi^2_{NC}(1, \frac{\gamma_j^2}{\tilde{s}_s(j)})$ according to [14]. Where the non-central χ^2 distribution is defined by the shift parameter $\beta_j = \frac{\gamma_j^2}{\tilde{s}_s(j)}$, and has mean and variance given by [14]:

$$\begin{aligned} E\left[\chi^2_{NC}\left(1, \beta_j\right)\right] &= 1 + \beta_j \\ Var\left[\chi^2_{NC}(1, \beta_j)\right] &= 2(1 + 2 \cdot \beta_j) \end{aligned} \tag{3.33}$$

Thus, in any histogram bin j where an anomaly is present, $\tilde{r}_s(j)$ and $\tilde{s}_s(j)$ are from different distributions and the corresponding term in the summation (3.25) is no longer $\chi^2(1)$ distributed but follow a non-central chi-square distribution $\chi^2_{NC}(1, \beta_j)$.

3.3.2.3 Relating Detection Parameters to the Distributions

Finally, we describe how the probability of false alarm, P_{FA}, and the probability of false negative, P_{FN}, relate to the central and non-central χ^2 distributions. These are plotted in Figure 3.5 for particular values of the degrees of freedom, which we denoted previously as N_{bins} and the shift parameter β. Let χ^2_D denote the chi-square score (computed using (3.25)) necessary to decide that $\tilde{r}_{s(i)}(j)$ and $\tilde{s}_{s(i)}(j)$ deviate sufficiently, statistically speaking, to indicate an anomaly. The value of χ^2_D is given by:

$$\chi^2_D = F^{-1}\left(\chi^2(N_{bin}), 1 - P_{FA}\right) \tag{3.34}$$

where F^{-1} is the inverse cdf of $\chi^2(N_{bin})$, the central chi-square distribution with N_{bin} degrees of freedom, evaluated at $1 - P_{FA}$. If $\tilde{r}_{s(i)}(j)$ and $\tilde{s}_{s(i)}(j)$ are from the same distribution, i.e., no anomaly present, the chi-square score obtained using (3.25) will only exceed χ^2_D with probability P_{FA}.

When $\tilde{r}_{s(i)}(j)$ and $\tilde{s}_{s(i)}(j)$ are from different distributions then the chi-square score follows a non-central chi-square distribution, i.e., $\chi_i^2 \sim \chi_{NC}^2(N_{bins}, \beta_i)$. The shift parameter, β_i of the χ_{NC}^2 distribution determines the shift left-to-right, of the distribution, and can intuitively be thought of a measure of how much $\tilde{r}_{s(i)}(j)$ and $\tilde{s}_{s(i)}(j)$ differ when computing (3.25). For a histogram bin, j, in the sub-density being tested, when $\tilde{r}_{s(i)}(j) \neq \tilde{s}_{s(i)}(j)$ then the difference, γ_j, is given by (3.32). Note that, if $\tilde{r}_{s(i)}(j)$ and $\tilde{s}_{s(i)}(j)$ are from different distributions, then the difference γ_j increases as more events are observed. The value of γ_j determines the shift parameter according to $\beta_{i,j} = \frac{\gamma_j^2}{\tilde{s}_{s(i)}(j)}$ [67]. Further, the combined shift parameter, β_i, for the sub-density being tested is the sum of the terms:

$$\beta_i = \sum_{j \in s(i)} \beta_{i,j} = \sum_{j \in s(i)} \frac{\gamma_j^2}{\tilde{s}_{s(i)}(j)} \tag{3.35}$$

The probability of a false negative, P_{FN}, is the $Pr(\chi_i^2 < \chi_D^2)$ when $\chi_i^2 \sim \chi_{NC}^2$, i.e., an anomaly is present. Once χ_D^2 is determined, P_{FN} is decreased by increasing the shift parameter, which shifts the distribution of χ_{NC}^2 towards ∞.

As an example to illustrate how P_{FA} and P_{FN} relate to the actual χ^2 distributions consider Figure 3.5, which plots the probability density functions for the two χ^2 distributions. The solid line is a centralized χ^2 distribution with 5 degrees of freedom, and the dashed line is a non-centralized χ^2 distribution with 5 degrees of freedom and a shift parameter equal to 10. When no anomaly is present in the measurements then the chi-square score χ_i^2, given by (3.25), follows the centralized $\chi^2(5)$ distribution (solid curve), and when an anomaly is present then χ_i^2 follows the non-centralized $\chi_{NC}^2(5, 10)$ distribution (dashed curve). In this example, the value, $x = 11.07$, corresponds to χ_D^2. P_{FA} is the area under the solid curve to the right of $x = 11.07$, and P_{FN} is the area under the dashed curve to the left of $x = 11.07$. Note, that increasing the value of the shift parameter, β, shifts the non-central χ_{NC}^2 distribution

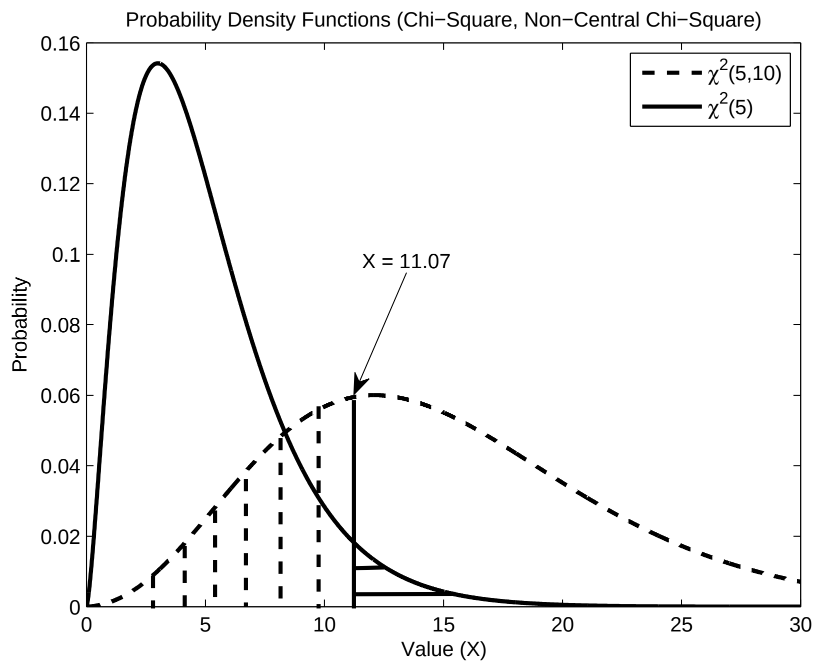

Figure 3.5: Example of Central and Non-Central χ^2 Distributions

towards the right in Figure 3.5, leading to less overlap between the distributions and consequently a lower chance for false alarms. The value of β can be thought of as the amount of evidence for an anomaly that has been collected, which is a function of the rate of the anomaly and the amount of time over which the events have been observed.

With explicit expressions for the distribution of χ_i^2 for both the null and alternative hypothesis it is possible to determine optimal parameters for the detection system, which will be discussed in Sections 4.3.1, 4.4.1.

3.3.3 Smooth Approximation

As mentioned in the previous section, the "smooth approximation" is a distribution designed to be anomaly free. We propose two methods: one for an ideal renewal process and one to be used for actual Internet traffic. The smooth approximation will be denoted $\tilde{s}(j)$, and we assume that the approximation will be chosen for the specific method, i.e., different approximations for an ideal renewal process and real traffic.

3.3.3.1 Smooth Approximation - Renewal Process

The first smooth approximation makes use of the fundamental property of the renewal function (and associated renewal density), which was described in (3.5), namely that given the distribution of the first order partial sum sequence, i.e., $F_1(M)$ or $f_1(M)$, all other orders can be determined via convolution. Therefore, the entire renewal function (or density) can be determined from $F_1(M)$ or $f_1(M)$. Similarly, the smooth approximation can be determined using the estimate of the inter-arrival pdf, i.e., $\tilde{f}_1(j)$, by taking multiple convolutions of $\tilde{f}_1(j)$:

$$\tilde{s}(j) = \sum_{k=1}^{\infty} \tilde{f}_k(j) = \sum_{k=1}^{\infty} \tilde{f}^{k*}(j) \tag{3.36}$$

where $\tilde{f}^{k*}(j)$ is the k-fold convolution of $\tilde{f}_1(j)$.

The smooth approximation assumes that $\tilde{f}_1(j)$ is not affected by the anomalous events. This assumption is reasonable given the fact that for the detection problem considered here the anomalous events are low-rate, i.e., P is large. Therefore, multiple background events occur between pairs of anomalous events meaning that the distribution $\tilde{f}_1(j)$ does not have a strong component at $j = P$ and consequently the convolutions of $\tilde{f}_1(j)$ do not have a strong component at $j = P$ either.

3.3.3.2 Smooth Approximation - Non-Renewal Process

The second method to derive a smooth approximation is necessary for traffic that does not follow a renewal process. In this case, the first order inter-arrival pdf cannot be used to compute the higher order inter-arrival pdfs via convolution. The failure of the convolution method to produce an accurate smooth approximation for Internet traffic is due to one issue in particular, so-called “multiple-packet deterministic sequences”, which are caused by packets being queued at routers then output from the router back-to-back [40]. The inter-arrival timing of the back-to-back packets is a function of their packet sizes, which is known to follow a roughly bi-modal distribution [71]. The deterministic nature of the inter-arrival times for back-to-back packets causes the inter-arrival time (or first order partial sum) distribution to deviate from what would be expected for a renewal process. More specifically, the authors of [40] state that the inter-arrival distribution consists of two portions: “one that can contain back-to-back packets and the other for packets that are guaranteed to be separated by idle time”.

While the “multiple-packet deterministic sequences” create deviations in the inter-arrival time (or partial sum) distributions for low orders, the length of each back-to-back packet sequence is relatively short. Because the deterministic sequences are short their impact on higher order partial sum distributions is less apparent. Herein lies the problem with attempting to estimate higher order partial sum distributions via the first order partial sum distribution, as in the first method. Taking the n-fold convolution of the first order partial sum distribution would generate an estimate of the n-th order partial sum distribution that is heavily biased towards having very long deterministic sequences, i.e., many consecutive back-to-back packets.

Since the anomaly could not be removed using the convolution technique we chose to apply a filtering method directly to the renewal density estimate, $\tilde{r}(j)$, itself. The filtering method selected is called the trimmed mean and is particularly suited for for computing the mean value of data even in the presence of outliers [72]. The trimmed mean is a technique that removes spurious data before computing the average value, which makes it possible for us to compute an anomaly free distribution from the renewal density estimate even when there are sharp peaks in the data near $t = P$.

To derive the smooth approximation, $\tilde{s}(j)$, of the renewal density the trimmed mean is computed using the renewal density estimate, $\tilde{r}(j)$, at each value of j. To compute the trimmed mean at a given j, we use the following algorithm on the the $\pm J$ neighboring values of $\tilde{r}$, i.e.,

$$\{\tilde{r}(j-J), \tilde{r}(j-J+1), ..., \tilde{r}(j-1), \tilde{r}(j+1), ..., \tilde{r}(j+J-1), \tilde{r}(j+J)\} \tag{3.37}$$

(note: that $\tilde{r}(j)$ is not used).

1. Order the neighboring values (either ascending or descending), such that $\{\tilde{r}(j - J(1)) > \tilde{r}(j - J(2)) > ... > \tilde{r}(j - J(2 \cdot T))\}$ for some ordering of $J(i) \in [-J, -1], [1, J]$.

2. Remove the top and bottom $Trim\%$ of the values.

3. $\tilde{s}(j)$ is the mean of the remaining values.

The length J is user-selectable, however, a good choice is to set $J = h \cdot N_{bin}/2$, i.e., compute the trimmed mean on an interval equal to the width of a sub-density. Typical values for $Trim$ are $5 - 25\%$ [72]. However, in practice the anomalous events may experience jitter and therefore can impact many neighboring values of j. Thus a larger value is used, $Trim = 40\%$, and has shown to provide a robust smooth approximation.

3.3.4 Jumping Windows

In Section 3.3.2 the operation of the detection method was described as it is applied to a particular sub-density. To implement the detection system shown in Figure 3.4 it is desirable that each sub-density be able to output detection decisions at some fixed interval, for example T_{Fix}. The problem with this scenario, however, is that each sub-density requires a different amount of time to gather enough measurements, including anomalous events, in order to determine with confidence that indeed an anomaly is present. We denote the amount of time required to gather enough anomalous events the time-to-detection, $T_D(j)$, where j indicates the sub-density. In Sections 4.3.1.1 and 4.4.1 we discuss how to determine $T_D(j)$ in various anomaly detection examples.

To implement our system such that each sub-density outputs detection decisions at a fixed interval we propose to use jumping windows with varying length. To obtain the variable length we group measurements in time windows equal to the detection decision reporting interval T_{Fix}. We call these time windows observation windows and denote them $W(i)$. Then each sub-density, j, uses a different number of observations windows, denoted $NumW(j)$, to perform detection on, which is a function of the time-to-detection, $T_D(j)$, in that sub-density and is given by:

$$NumW(j) \geq \frac{T_D(j)}{T_{Fix}} \tag{3.38}$$

The jumping window implementation is shown in Figure 3.6. For each observation window, $W(i)$, an estimate of the renewal density is generated using the inter-arrival times computed from the measurements that arrive during this window. In other words only partial sum sequences, $S_n(M(j))$, are computed where $M(j) \in W(i)$, the current observation window. Note that measurements from previous windows, i.e., $M(j-n) \in W(i-1)$, are used in the computation of these inter-arrival times, however, the starting measurement must occur within the current observation window.

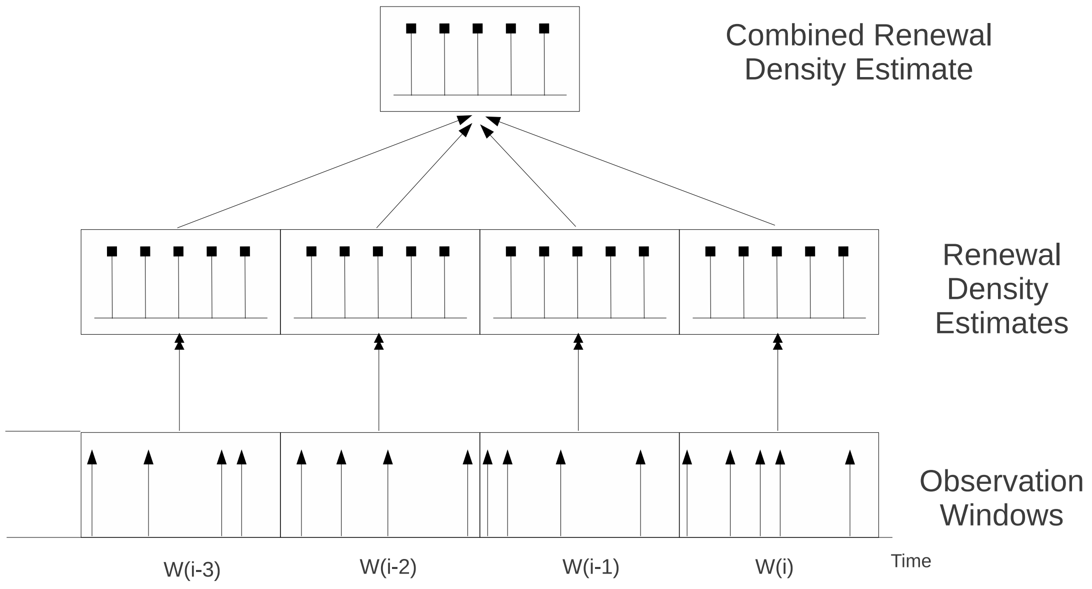

Figure 3.6: Jumping Window Implementation of Detection System

When the current observation window ends, signaling the system to output detection decisions, detection is performed in sub-density j by taking the sum of renewal density estimates from the past $NumW(j)$ observation windows. For example, in Figure 3.6 renewal density estimates from the last four observation windows are summed to create the combined renewal density estimate that is used in the Pearson Chi-Square test.

When an observation window is no longer needed for detection the measurements and renewal density estimate are forgotten, and replaced by data from the next observation window.

3.3.5 System Parameter Selection

In this section methods for selecting some of the detection system parameters are discussed in a general context, and in Sections 4.3 and 4.4 system parameter selection will be discussed as it applies to specific detection problems. The detection system parameters to be determined, which are shown in Figure 3.7, include: $Maxx$ - maximum inter-arrival time, h - histogram bin width, H - sub-density width, and $Nsub$ - number of sub-densities.

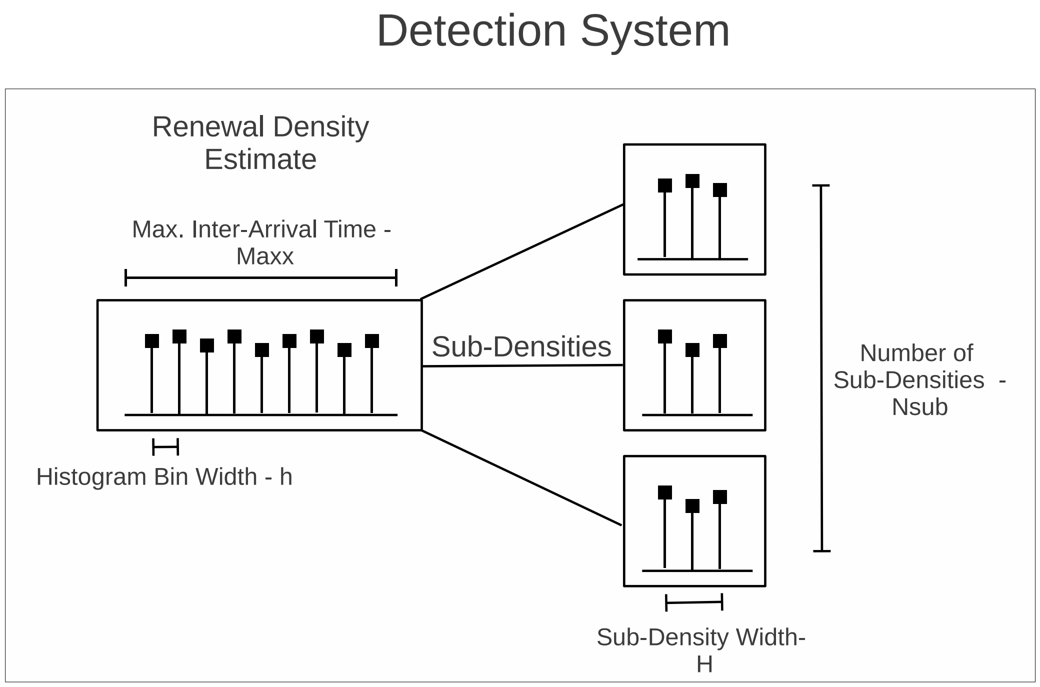

Figure 3.7: Selecting Detection System Parameters

3.3.5.1 Width of Sub-Density

As stated in Section 3.3.1, detection is performed independently in sub-densities. Selecting the proper width of the sub-density, H, is important because it affects detection performance. For example, if the width of the sub-density is selected too large then the impact of the periodic anomaly, which we assume occurs at or near $t = P$, on the Chi-Square score will be lessened. This occurs because the surrounding histogram bins do not contain the anomaly, and the Chi-Square score in histogram bins not containing the anomaly is approximately zero, as indicated by (3.25). In practice, however, the anomalous events are not exactly periodic due to jitter created by queuing in routers as the packets traverse the Internet. Because of jitter the peak in the renewal density caused by the periodic events does not occur only at $t = P$. Instead the peak is spread over a range of inter-arrival times $t = P \pm \Delta$, where Δ is determined by the amount of jitter encountered in the network. If the periodic events are jittered we must select the width of the sub-density such that it captures as much of the periodic events as possible.

Therefore, in a general context the optimal size for the width of the sub-density, H, should be as small as possible as long as it captures the majority of the anomaly in a single sub-density. We define the width of the anomaly, H_A, to be the range of inter-arrival times that the anomaly impacts in the renewal density. The range of inter-arrival times impacted depends on the type of anomaly being detected. For example, in the jitter scenario discussed above $H_A = 2 \cdot \Delta$ (estimating the value of Δ is discussed in Section 4.2.3), or for the so called shrew attack, discussed in Section 4.4, H_A depends on the burst interval of the attack. In Sections 4.3 and 4.4 we discuss methods to determine H_A for the detection examples considered.

In a practical scenario we recommend that the width H be selected to be two to four times H_A because we cannot know that the anomaly is centered in a sub-density and this added margin ensures the majority of the anomaly is captured.

3.3.5.2 Histogram Bins per Sub-Density and Width of Histogram Bin

In addition to the parameters shown in Figure 3.7 one additional parameter, N_{bin}, which is the number of histogram bins per sub-density also must be determined. N_{bin} is related to H and h by, $H = h \cdot N_{bin}$. Since H has already be determined we are left with selecting h or N_{bin}. Which one of the two parameters is determined first is a matter of practicality.

Theoretically it is easier to select h using (3.22) or (3.23) to guarantee convergence of the histogram bins. However, in practice it is easier to select H and N_{bin} and let h be determined. Selecting H and N_{bin} is easier due to the way the system is implemented. As described in Section 3.3.2, to reduce computational cost, the selection of a hypothesis, whether an anomaly is present or not, is done by comparing the value of the chi-square score computed, χ_i^2, against the threshold, χ_D^2. The value of the χ_D^2 is determined by P_{FA} and N_{bin}. There is no simple closed form expression for χ_D^2 in terms of P_{FA} and N_{bin}. Instead for the system implementation, values of χ_D^2 are pre-computed for a selection of P_{FA} and N_{bin} combinations. Thus, because χ_D^2 is precomputed it limits the possible values of N_{bin}, and because of this in a practical system we select the value of N_{bin} and allow the value of h to be determined.

To ensure that the value of h, determined by $h = H/N_{bin}$, meets the criteria for convergence given in Section 3.2.3.2, we pre-compute the value of χ_D^2 with multiple values of N_{bin} for each value of P_{FA} we desire to use. Then when conducting an experiment, based on the characteristics of the Internet traffic we select the value of N_{bin} such that h meets the convergence criteria given by (3.23).

3.3.5.3 Maximum Inter-Arrival Time

Selecting the maximum inter-arrival time depends on the the maximum period of the anomalous events being detected. Clearly, $Maxx > P$, is required in order for detection to be possible. Since the value of P is not know *a priori* the user can select $Maxx$ based on the period of anomaly they determine important. For example, from a system administrator perspective, on a network with a capacity of 10 Gbps a 1 Mbps periodic anomaly might not be a concern, however, on a 100 Mbps network a 1 Mbps anomaly is a major concern. Therefore, from a practical perspective selecting the value of $Maxx$ should be done by the system administrator based on what period of anomaly is important to detect on their system. If we define, PS_{Avg}, to be the average packet size in the network (which can be easily determined using standard system administrator tools), then we can relate P to the rate of the anomaly, R Mbps by, $P = \frac{PS_{Avg} \cdot 10e6}{R}$. Finally, the system administrator can select $Maxx$ using either, $Maxx > P$ or $Maxx > \frac{PS_{Avg} \cdot 10e6}{R}$.

Selecting $Maxx$ using this criterion works for the general periodic anomaly detection scenario. However, assuming the detection system can maintain the computation and data storage burden it is desirable to select $Maxx \gg P$ to detect harmonics of the fundamental anomalous period. In particular, in Section 4.3.3, we show that in order to detect quasi-periodic anomalies, when the fundamental period is obscured due to a smart-adversary, detecting harmonics is a more reliable and potentially the only feasible method to detect the anomaly.

3.3.5.4 Maximum Inter-Arrival Order

The theoretical method to select the maximum inter-arrival order is to select n such that $F_n(Maxx) < \phi$, where F_n is the cdf of the n-th order inter-arrival time distribution and ϕ is a small value which indicates the probability of an n-th order inter-arrival occurring before $Maxx$ to be negligible.

In practice, determining the maximum inter-arrival order to be computed between measurements in order to estimate the renewal density can be done by assuming a Poisson process model for the Internet traffic measurements. Given the Poisson process model the first order inter-arrival times, $f_1(t)$, are exponentially distributed with mean inter-arrival time λ. Under the same assumption the distribution of the n-th order inter-arrival times, $f_n(t)$, given by (3.5) is the n-th order convolution of the first order inter-arrival distribution and follows an Erlang distribution with scale parameter $n \cdot \lambda$.

To determine the value n such that $F_n(Maxx) < \phi$ exactly requires that we compute the cdf of the Erlang distribution, which is impractical because it involves computing the factorial of n. Instead of computing the exact cdf of the Erlang distribution we use the Normal approximation to the Gamma distribution that we used in Section 3.2.3.2, i.e., for large n the pdf of the Erlang (or Gamma) distribution (with scale parameter λ) is approximated by, $f_n(t) \sim N(n \cdot \lambda, n \cdot \lambda^2)$. Using the Normal approximation the cdf can be expressed in terms of the error function, erf, by the following well known expression [30]:

$$F\left(x, \mu, \sigma^2\right) = \frac{1}{2}\left[1 + erf\left(\frac{x-\mu}{\sigma\sqrt{2}}\right)\right] \tag{3.39}$$

and inserting the values from our Normal approximation:

$$F\left(Maxx, n \cdot \lambda, n \cdot \lambda^2\right) = \frac{1}{2}\left[1 + erf\left(\frac{Maxx - n \cdot \lambda}{\lambda\sqrt{2 \cdot n}}\right)\right] \tag{3.40}$$

Next, we can express our constraint condition $F_n(Maxx) < \phi$ using (3.40) as:

$$erf\left(\frac{Maxx - n \cdot \lambda}{\lambda\sqrt{2 \cdot n}}\right) < 2 \cdot \phi - 1 \tag{3.41}$$

The user then selects a desired value for ϕ and using a table that gives values for $erf(\frac{\theta}{\sqrt{2}})$, where in our scenario:

$$\theta = \frac{Maxx - n \cdot \lambda}{\lambda\sqrt{n}} \tag{3.42}$$

the value of n can be determined. Tables of values for erf can commonly be found in probability references such as [30].

The selection of n is important because computing the inter-arrival times between measurements, which is done up to order n, is the computational bottleneck in our detection method. Discussed in detail in Section 5.1.2, the computational cost of our detection system is on the order of n^2. Therefore, to minimize computational costs selecting n as small as possible is desirable. If computational cost is not a concern, then less precise yet simpler methods to select n can be used. For example, using the Erlang approximation used above, the average inter-arrival time at order n is $n \cdot \lambda$. Selecting n such that $n \cdot \lambda > k \cdot Maxx$, where k is some integer multiple (e.g., $k = 5$ or $k = 10$), would be one simple method to select the value of n.

3.4 Conclusion

In this chapter the design of a low-rate periodic event detection system, IA^{2}D, was discussed. IA^{2}D uses an estimate of the renewal density, which it breaks down into segments called sub-densities that are processed individually yet by using jumping windows of varying length all sub-densities report detection decisions at a fixed user defined time interval, T_{fix}. In the following chapter, IA^{2}D will be used in two different detection scenarios. Detecting low-rate periodic events in Internet traffic, resembling a DoS attack, using IA^{2}D is the subject of Section 4.3, and in Section 4.4 the task of detecting Shrew attacks is attempted using IA^{2}D.

Chapter 4:

Detection Examples using IA²D

4.1 Introduction

In this chapter we consider detection of periodic events in aggregate collections of measurements. More specifically we consider detection of very low-rate periodic packet arrivals mixed with random background Internet traffic. These periodic packet arrivals potentially represent malicious behavior, e.g., denial-of-service (DoS) attack traffic [34, 36] or benign yet undesirable network behavior like under-performing or "bottleneck" links [9, 41, 77].

Bottleneck links, as shown in Figure 4.1, output packets back-to-back at their maximum rate, R_1. The time, say P, between back-to-back packets is determined by the link rate and the packet size, L, i.e., $P = L/R_1$ [48]. According to [71] the majority of packets flowing through the Internet have packet sizes of one of few dominate sizes. Therefore, in a bottleneck link with limited output rate R_1, the majority of packets are output back-to-back and the inter-arrival times between packets correspond to the P times associated with these common packet sizes. Further, even when the back-to-back packets flow into a network with a higher link speed, the time between the packets will remain equal to P even though the packets are no longer back-to-back.

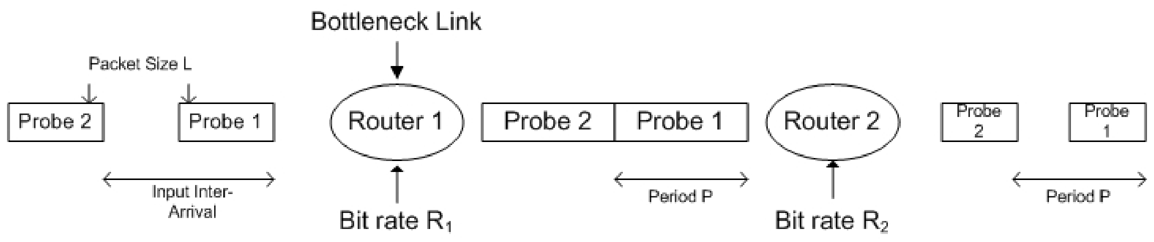

Figure 4.1: Bottleneck Link Creates Periodic Inter-Arrival

Like bottleneck links, packets from a DoS attack can arrive at the target periodically spaced. There are two main scenarios where DoS attack traffic can manifest itself as periodically spaced traffic. The first scenario occurs due to limitations in the system generating the DoS attack. For example, the algorithm used in the attack outputs packets at a fixed period. Alternatively, the system used in the attack attempts to output packets as fast as possible, yet the host operating system or central processing unit limits the output rate causing the attack packets to be output periodically [35]. While DoS attacks may not typically be low-rate, consider the case of an individual stream of attack packets in a Distributed Denial-of-Service (DDoS) attack. For a DDoS attack the combination of multiple attack streams might be noticeable near the target, however, at other points in the network the individual streams are at a lower rate and are less obvious. It is the goal of our system to detect these low-rate periodic anomalies. The second DoS scenario that produces periodic anomalies is called low-Rate, TCP-targeted DoS attacks (so called shrew attacks) [46], a recent addition to the arsenal of DoS attacks. The goal of the shrew attack is not to overwhelm the target, but to interfere with the targets normal operation by disrupting TCP congestion avoidance algorithms. This interference negatively impacting the throughput of the target, and represents a potential loss of income in commercial systems.

In this chapter we apply the IA^{2}D detection system that we designed in Chapter 3 in two different detection scenarios. The first scenario (Section 4.3) is to detect very low-rate periodically spaced packet arrivals that are merged with background Internet traffic. The second scenario considered (Section 4.4) is detecting a low-rate, TCP-targeted shrew attack.

These applications of our detection system provide good opportunities to showcase the use of a Poisson model of Internet traffic that we use to analyze the detection system and help in selecting actual measurement system parameters. Using the Poisson model we can derive theoretical values for the system parameters. Further, because the applications are disparate it shows the applicability of our system to a variety of detection scenarios. In particular in Sections 4.3.1 and 4.4.1 we analyze our detection system using a Poisson process for both detection scenarios in order to determine the time-to-detection, T_D. Recall from Section 3.3.4 that T_D indicates the time required to gather enough measurements, anomalous events, to determine with confidence that indeed an anomaly is present. Thus, by computing T_D we know the required observation window length that must be considered when performing detection.

While the system analysis using the Poisson process is helpful to determine T_D, in practice Internet traffic measurements do not follow a Poisson process. Therefore, in Sections 4.3.2 and 4.4.2 we consider methodologies to modify the system parameter settings determined using our Poisson process analysis in order to make IA^{2}D perform reliably on real Internet traffic. Two modifications are used in particular. The first is the use of subsampling to make the Internet traffic measurements more like the Poisson process. Why subsampling works to make Internet traffic more like a Poisson process is described in Section 4.2.1. The second modification is to the value of T_D, where we recommend the use of what we call the "worst case scenario" value of T_D. The worst case value of T_D accounts for the fact that while traversing the Internet the periodic traffic encounters delays at routers, which cause the inter-arrival times between consecutive periodic packets to be jittered. The jitter of the periodic inter-arrival times causes the peak in the renewal density near $t = P$ to be smoothed. Thus the difference γ_P in (3.32) is decreased, and a longer T_D is required in order to achieve the χ^2 score need for detection. Computation of the worst case values of T_D are described in Sections 4.3.1 and 4.4.2.

Using these modifications we show in Sections 4.3.2 and 4.4.2 that we can reliably detect the two different types of periodic traffic when merged with real Internet traffic. The experiments conducted here are designed to verify that the system parameters selected using the Poisson process analysis produce reliable results, in terms of P_{FA} and P_{FN}. In a practical system it is desirable to limit user interaction, e.g., checking whether a positive detection is due to an actual anomaly or a false positive. Methods to reduce user interaction, including post-processing of positive detection results and reducing P_{FA} by increasing T_D, are discussed in Section 4.5.

4.1.1 Related Work

A number of different techniques have previously been applied to the task of detecting periodic packet arrivals in Internet traffic. Most methods use a time series representation for the Internet traffic measurements, where the signal represents the number of packet arrivals in uniform time intervals. Working with the time series representation Katabi *et al.* used wavelets to detect traffic congestion due to bottleneck links in the network [41]. Similarly, in [34] He used spectral based techniques to detect bottleneck links using aggregate network traffic measurements. Both of these methods successfully detected periodic traffic caused by bottleneck links, however, the periods that were considered were much shorter than those we focus on in our work, i.e., higher rate problems were studied. In order for these methods to be able to detect the longer periods that we focus on, the length of the observation window upon which their time-frequency transform is performed would have to be increased. The increased window length is required because for longer periods the length of time needed to observe the same amount of periodic events, as for the shorter periods, increases. Computing the time-frequency transforms used in [41] and [34] on the increased window length significantly increases computational complexity. The benefit of our method over these methods is that the computation cost is determined by the maximum inter-arrival order n used to estimate the renewal density, and therefore we can increase the observation window length without increasing computation cost.

The point process signal representation that we use has also been used to detect anomalies in Internet traffic. In [42], Katabi *et al.* analyzed the entropy of packet inter-arrival times in order to detect congestion at network links. In congested links the inter-arrival times take on only a few values (related to the packet sizes) and have a low entropy. Instead, in uncongested links the inter-arrival times are random and have a higher entropy. The congested link in this example is similar to a bottleneck link and the packets output from a congested link have inter-arrival times that are essentially periodic. The main difference between this method and ours is that this method only looks at first order inter-arrival timing information. Thus, this method cannot be used to detect low-rate periodic packet arrivals in aggregate traffic because, as we assumed, the periodic packet arrivals are rarely back-to-back. In another work, Ma and Hellerstein [49] used the point process signal representation to detect periodic packet arrivals in Internet traffic. This method is similar to ours in that it applies the Pearson Chi-Square test to the distribution of inter-arrival times. However, as in Katabi's work, Ma and Hellerstein only consider the first order inter-arrival information. Again this method would not work for our task of detecting low-rate periodic packet arrivals because the periodic packets do not occur back-to-back in Internet traffic. Further, this method uses an exponential distribution to model the first order inter-arrival times for comparison in the Pearson test, which as we will show in Section 4.2.1 is not a reliable model for real Internet traffic.

The detection method that is most similar to ours, in terms of application, is the bivariate Parametric Detection Mechanism (bPDM) [75]. bPDM was designed to detect constant rate anomalies even at very low rates, where constant rate means that there is a constant number of anomalous packets per unit time. bPDM is a model based method that uses a time series signal representation, and analyzes the incoming traffic measurements to determine whether or not the traffic is random or random plus a constant component. Constant rate includes periodic anomalies, therefore, this method works on a more general class of anomalies compared to IA^{2}D. However, our method has the additional benefit that it can detect and *distinguish* between multiple periodic events that occur in the same set of measurements. Since bPDM is rate based it can detect the multi periodic events as a combined anomaly, however, it is unable to distinguish between the two periods. Being able to distinguish different periodic events is desirable because real Internet traffic does contain some inherent periodicities. For example, the round trip time to transmit packets between two computers is typically a fixed time (assuming the network is not highly congested). When two computers are in the middle of a session, e.g., streaming a video from a movie web site, packets being sent between the two computers can create a periodic stream, which clearly is not an anomaly. Therefore, it would be desirable to be able to determine whether a periodic stream is legitimate (not requiring mitigation), and still be able to detect additional, potentially malicious periodic traffic streams. This is possible with IA^{2}D and not with bPDM.

Because bPDM is the only technique we have found in literature capable of detecting periodic events at the low-rates we consider, it is the only method we use for comparison to our method (see Section 4.3).

4.2 Preliminaries

Before proceeding to the applications of the IA^{2}D detection system we present some preliminary analysis and parameter selection that is universal to all scenarios.

4.2.1 Renewal Process Assumption

Modeling Internet traffic as having independent arrivals or as a Poisson process is a debated point. For example, in [58] Paxson shows that a Poisson model for Internet traffic fails to capture the self-similarity of Internet traffic at very long time scales. However, we support the use of a renewal process model based on the result from Karagiannis [40], which states that "(p)acket arrivals appear Poisson at sub-second time scales," assuming that measurements are recorded at systems seeing "a vast number of different multiplexed flows." The low-rate detection task we consider in our work coincides with the observations in [40] because we examine aggregate traffic so our measurements represent a collection of multiplexed flows, and in most scenarios we consider sub-second time scales. For example, in Section 4.3 the periods we detect are on the order of milliseconds, and is Section 4.4 the periods are larger, in the range of .5 to 2 seconds. We note that even at longer time scales, or even when the renewal process model does not hold the techniques we use, i.e., estimating the renewal density, etc., are still valid. However, the analysis we perform using Poisson process tends to underestimate some of the system parameters that we use the analysis for. Therefore, we scale these estimates based on characteristics of the traffic in order for our detection system to achieve the desired P_{FA} and P_{FN} values.

Even if the Poisson process model is adopted there may be problems with "multiple-packet deterministic sequences", created by back-to-back packets, which causes the packet inter-arrival distribution to deviate from an exponential distribution [40]. More specifically, the authors state that the inter-arrival distribution consists of two portions: "one that can contain back-to-back packets and the other for packets that are guaranteed to be separated by idle time." In their analysis they found that the inter-arrival distribution deviated from the exponential distribution at short inter-arrival times because of the back-to-back packets. However, they found that this deviation was small in networks where the links were not heavily utilized.

For our work these "multiple-packet deterministic sequences" create false periodic peaks in the renewal density, and were a significant cause of false positives. To combat the deterministic sequences we use subsampling and back-to-back packet filtering to reduce the effect of these sequences, i.e., remove the periodic peaks they create in the renewal density.

4.2.2 Subsampling and Back-to-Back Packet Filtering

In this section we present two methods to reduce the effect of "multiple-packet deterministic sequences" and to make Internet measurements more like a Poisson process. The first method is related to the result from Rényi discussed in Section 3.2.1.1, which stated that the process obtained from thinning a general renewal process converges to a Poisson process as the thinning rate goes to ∞. Relating this idea to that of Karagiannis *et al.* (from the previous section), if we subsample (or thin) Internet traffic measurements we remove many of the back-to-back inter-arrival times leaving primarily inter-arrivals which are separated by an amount of idle time. Thus, subsampled Internet measurements are actually more accurately modeled by a Poisson or renewal process. The second method more specifically addresses the problem of "multiple-packet deterministic sequences" by removing, or filtering back-to-back measurements. This method is quite a useful alternative to subsampling in scenarios when a low subsampling rate is desired because it can also produce "Poisson-like" measurements, but with lower subsampling rates. A lower subsampling rate is desirable because subsampling increases the time required to gather enough periodic events in order to detect the presence of an anomaly.

In order to show the effect of subsampling on Internet traffic measurements, in terms of making the measurements more like an ideal Poisson process, we performed the following experiment. We begin with typical Internet traffic, which has non-Poisson characteristics (as explained previously). These characteristics impact the renewal density that we estimate from the background traffic, and consequently impacts our detection reliability. By applying back-to-back packet filtering or subsampling we improve the fit of our measurements to a Poisson distribution, and improve the reliability of our detection system with the associated trade-off of a longer T_D. Thus, in our experiment we randomly subsample, using our subsampling technique derived in Section 2.3, a typical Internet traffic measurement trace [1] at multiple rates, $\mu = \{1, 5, 10\}$. For each subsampled trace inter-arrival distributions, $\hat{f}_n(t)$, at orders $n = [1 : 500]$ are estimated using the method in Section 3.2.3. For each order n the average inter-arrival time λ_n is computed, and an Erlang distribution, $Erl(n, \lambda_n/n)$ is generated. Finally, the KL divergence is computed between the Erlang distribution and the estimated inter-arrival distribution at each order n and for each subsampling rate μ. Smaller values of the KL divergence indicate that the estimated distribution $\hat{f}_n(t)$ are more like the $Erl(n, \lambda_n/n)$ distribution, which is what a Poisson process would generate at order n.

The values of the KL divergence are shown in Figure 4.2 and clearly show that as subsampling is increased inter-arrival time distributions estimated from the resulting measurements are indeed more like those of a Poisson process. This result coincides with the result of Rényi from thinning a renewal process and will become important in Section 4.3.2 when we attempt to use IA^{2}D on measurements obtained from a real Internet measurement system. In particular we see that the largest improvement (i.e., biggest differences in KL divergence compared to no subsampling) occur at large inter-arrival orders. This is beneficial for our detection problem because we try to detect low-rate events, which typically occur at higher inter-arrival orders. At low inter-arrival orders the presence of the "multiple packet deterministic sequences" still exists causing the KL divergence values at these orders to remain large compared to the value at higher inter-arrival orders.

Next, to show that by removing measurements corresponding to back-to-back packets it is possible to achieve 'Poisson-like' measurements with a lower subsampling rate we repeated the above experiment and for comparison show the KL divergence of measurements generated using a few subsampling rates, $\mu = \{1, 1.2\}$. The value of 1.2 is selected because it equaled the equivalent subsampling rate of the back-to-back filtered measurements, for this experiment. Removing the back-to-back measurements requires knowledge of the packet size of the measured packets. Assuming this information is known, back-to-back packet filtering works as follows, here $PackSize(k)$ is the size of the kth packet in bits, and $BitRate$ is the bit rate of the incoming network link in bits per second bps. Note, that the timestamp of the filtered measurement is the *first* measurement in the collection of back-to-back measurements. We make this choice because the first measurement is the only one which reflects the actual arrival time of a packet; the other timestamps are determined by the packet size.

The KL divergence results comparing back-to-back packet filtering to subsampling

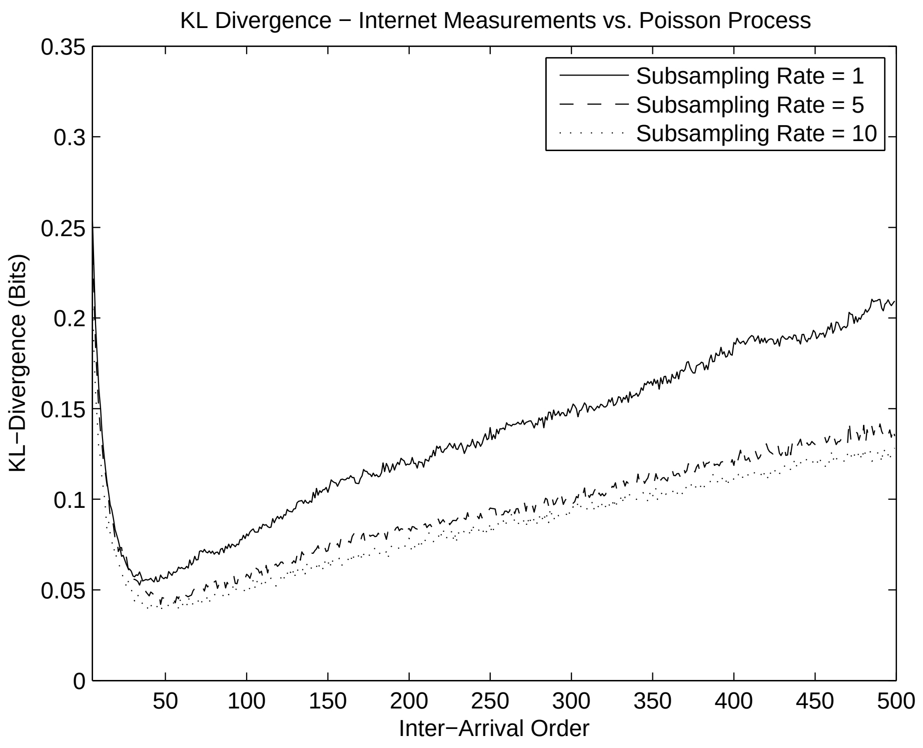

Figure 4.2: KL divergence results comparing subsampled Internet measurements to ideal Poisson process

are shown in Figure 4.3 (note that the maximum inter-arrival order is less in Figure 4.3 than in Figure 4.2). In this example we focus on the lower inter-arrival orders to show that by removing back-to-back packets there is a large improvement in KL divergence, especially at the lower inter-arrival order, between $5 - 20$ in the figure. The difference in KL divergence between non-subsampled measurements and the back-to-back packet filtered ones correlates with the idea from Karagiannis *et al.* that multi-packet deterministic sequences cause deviations from a Poisson approximation in inter-arrival time distributions from Internet traffic measurements. For comparison, the KL divergence of measurements subsampled by the equivalent rate, 1.2, is also shown, and we see that the result due to back-to-back filtering is an improvement over standard subsampling. This indicates that by using back-to-back packet filtering

Algorithm 1 Back-to-Back Packet Filtering

```
Require: input: measurements X(k) = M(k), 1, PackSize(k)
  loop
    i = 0
    while M(k+(i+1)) ¡ (M(k+i)+PackSize(k+i) * BitRate) do
      i++
    end while
    Y(n) = {M(k), i};
    k = k + i + 1
  end loop
  return measurements Y(n)
```

we can still obtain Poisson like measurements, yet because of the lower subsampling rate we do not have to sacrifice our time-to-detection as much as with standard subsampling. In particular we notice that the back-to-back packet filtering has a substantial gain over standard subsampling at lower inter-arrival orders indicating that it does a better job of removing the "multiple packet deterministic sequences". Therefore, if we have access to the packet lengths that are required for back-to-back packet filtering this method is more practical than standard subsampling.

4.2.3 Packet Delay Variation

For periodic detection, although it is assumed that the attack packets are at one point periodic, the task is to detect them in aggregate traffic. This involves merging the periodic events with background Internet traffic, which does not preserve the exact periodic structure of the attack packets. The merging process is done by routers which are only able to output one packet at a time. Packets received while the router is busy outputting another packet are queued and serviced by the router at the earliest possible time. If an attack packet is queued while the router processes a background Internet packet the periodic spacing between that attack packet and the attack packets before and after it will be altered. Merging attack and background traffic creates

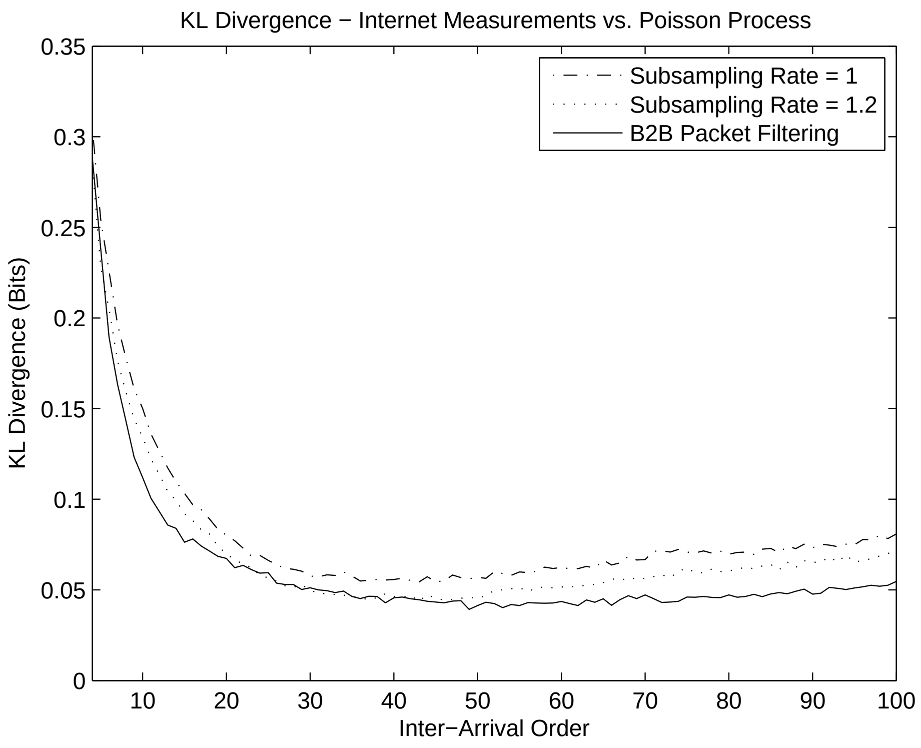

Figure 4.3: KL divergence results showing use of back-to-back packet filter to improve Internet measurements

variations in the periodic structure of the attack traffic, complicating the detection process. This variation is generally called packet delay variation, however, throughout this chapter we use the term jitter interchangeably. Methods to estimate the packet delay variation exist, see [22] for example, and we assume that when the periodic detection system is implemented a measure of the nominal packet delay variation is known or can be estimated. Assuming this quantity is known we can modify our detection system, by adjusting the time-to-detection T_D in order to take into account such delays. One simple method to estimate the packet delay variation between a given source (SRC) and destination (DST) is based on the following process [22]:

1. SRC sends packet to DST with timestamp indicating when packet was sent

2. DST receives packet, records timestamp when packet was received, and computes packet delay (PD) by taking the difference between timestamps ($TS_{rec} - TS_{sent}$)

3. Repeat 1 and 2 to compute multiple PD values, and generate packet delay variation (PDV) statistics from the PD values

In our model, assume that jitter will impact the periodic packets by delaying them by an amount of time that is uniformly distributed in the interval $[0, AvgJitter]$. Therefore the jitter causes the peak in the renewal density at $t = P$ to be spread out in a triangular distribution around the period, with maximum deviation of $\pm AvgJitter$.

4.3 Periodic Event Detection

Our first detection scenario is detecting periodic events, packet arrivals, in aggregate Internet traffic measurements. Figure 4.4 shows an example of this scenario, where we can see that the inter-arrival time between consecutive anomalous events is equal to P. However, because the background traffic is non-deterministic the number of background packets between pairs of consecutive periodic packets is random. Therefore the periodic inter-arrival times appear in the higher order inter-arrival distributions, i.e., the $f_n(t)$, at many different orders n. Further, we cannot simply search for periodic activity in one, or a few inter-arrival distributions, i.e., a specific $f_n(t)$, as the amount of periodic inter-arrivals in each would be small. However, with our renewal density based detection system it is possible to search within all inter-arrival distributions, $f_n(t)$ for all n, simultaneously. The presence of a periodic anomaly manifests itself in the renewal density, or the estimate, as peaks located at inter-arrival times corresponding to the period of the anomaly, P, and at harmonics of the fundamental

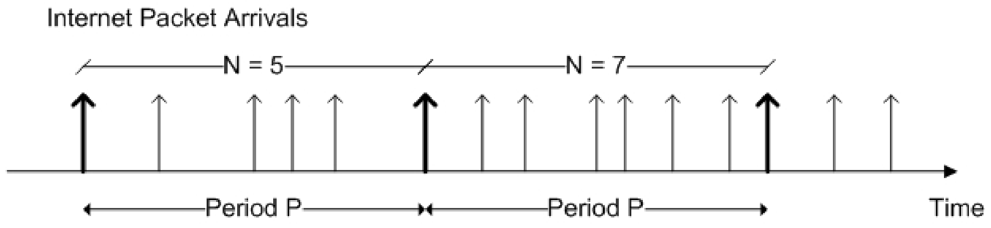

Figure 4.4: Example of Periodic Events Mixed with Other Measurements

period as well, i.e., $k \cdot P$; something we will exploit in Section 4.3.3. Figure 4.5 shows an example renewal density estimate with peaks created by a periodic anomaly at multiples of one second.

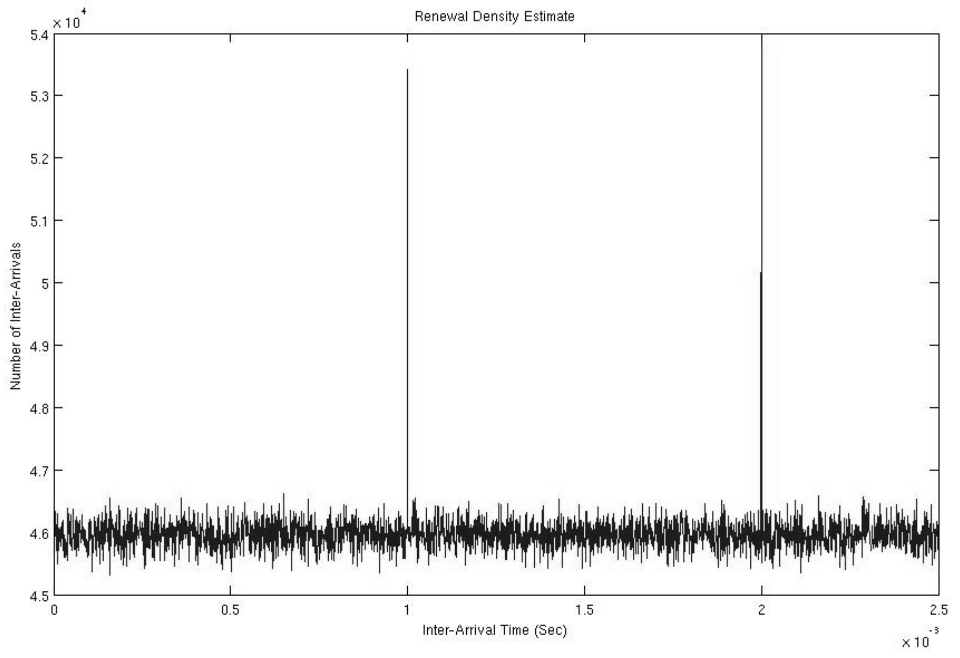

Figure 4.5: Example of Renewal Density Estimate with Periodic Event Peaks

We begin this section by continuing our analysis (from Section 3.3) of the system using Poisson processes, and from this analysis we derive an expression for the time-to-detection T_D. We compare the theoretic results against the bivariate Parametric Detection Mechanism (bPDM) [75], to show that our system is comparable to the state of the art. Further, we show an additional benefit that our method can detect and *distinguish* multi-period attacks; something bPDM cannot do.

Then, starting in Section 4.3.2 we apply our system to measurements of real Internet traffic. Some modifications to the T_D calculation are required to achieve the desired performance, however, following these adjustments our system is capable of reliably detecting very low rate periodic anomalies.

4.3.1 System Analysis with Poisson Process

In this section we continue our analysis of the detection system, from Section 3.3, using a Poisson process in to determine an explicit expression for the time-to-detection, T_D Intuitively, T_D, can be thought of as the length of time required until enough evidence, i.e., anomalous events, is gathered to decide whether an anomaly is present for a given P_{FA} and P_{FN}.

4.3.1.1 Determining Time-to-Detection

In order to determine T_D we use two of the parameters that were determined in Section 3.3.2. The first parameter we need is χ^2_D from (3.34), which is the chi-square score necessary to decide that $\tilde{r}_s(j)$ and $\tilde{s}_s(j)$ deviate sufficiently, statistically speaking, to indicate an anomaly. The second parameter is β_i from (3.35), which is the shift parameter of the χ^2_{NC} distribution. Recall, that when $\tilde{r}_{s(i)}(j)$ and $\tilde{s}_{s(i)}(j)$ are from different distributions then the chi-square score follows a non-central chi-square distribution, i.e., $\chi^2_i \sim \chi^2_{NC}(N_{bins}, \beta_i)$.

Next, to determine T_D we use the above parameters to compute β_D, which is the shift in the χ^2_{NC} distribution required to achieve a desired P_{FN} as described in Section 3.3.2. The value of β_D is given by,

$$\beta_D = \min_{\beta_D} \left(P_{FN} > F\left(\chi^2_{NC}(N_{bin}, \beta_D), \chi^2_D \right) \right) \tag{4.1}$$

i.e., β_D is the minimum shift parameter such that the cdf of χ^2_{NC} with N_{bin} degrees of freedom, evaluated at χ^2_D, is less than P_{FN}.

To determine an explicit expression for T_D in terms of β_D requires an exact expression for the renewal density. For this we assume that the background traffic is Poisson, and that combining a Poisson process with periodic events does not significantly impact the statistics of the Poisson process, i.e., it remains a Poisson process, but with a new inter-arrival time parameter. This assumption is valid for low-rate periodic events, such that multiple Poisson events occur between periodic ones. For a Poisson process with mean inter-arrival time λ combined with periodic traffic of period P, the new combined mean inter-arrival time $\hat{\lambda}$ is:

$$\hat{\lambda} = \frac{\lambda \cdot P}{\lambda + P} \tag{4.2}$$

As mentioned in Section 3.2.2 for a Poisson process, with mean inter-arrival time $\hat{\lambda}$ the renewal density is simply equal to $r(t) = 1/\hat{\lambda}$. Similarly, for the same Poisson process the value of the renewal density estimate $\tilde{r}(j)$, using histogram bins h seconds wide, is:

$$\tilde{r}(j) = \frac{h}{\hat{\lambda}} \tag{4.3}$$

For the Pearson's Chi-Square test we need to estimate the number of observed inter-arrival times equal to t, or more precisely the number of inter-arrivals that fall in the jth histogram bin, where the jth histogram bin includes all $t \in \{j \cdot h, (j + 1) \cdot h\}$. This is computed by multiplying the probability of such an inter-arrival time occurring, given by $\tilde{r}(j)$, by the total number of observed inter-arrival times, which for T seconds of measurements is:

$$E[\tilde{r}(j)] = \frac{T}{\hat{\lambda}} \cdot \frac{h}{\hat{\lambda}} = \frac{T \cdot h}{\hat{\lambda}^2} \tag{4.4}$$

Further, using method 1 (from Section 3.3.3.1) to derive the "smooth approximation" for an ideal renewal process (Section 3.3.3.1) we assume that $\tilde{s}(j) = E[\tilde{r}(j)]$, therefore, the value of $\tilde{s}_{s(i)}(j)$ is constant and given by (4.4).

The value of γ_j depends on the period of the anomalous traffic, which for this analysis is assumed known and equal to P. Assuming none of the periodic events are delayed or jittered when combined with the renewal process, then all of the inter-arrival times due to the periodic events are exactly equal to P. In this case the difference, γ_j between the renewal density estimate and the smooth approximation is zero except in the histogram bin containing the period P, i.e.,

$$\gamma_j = \begin{cases} 0 & \text{for } j \neq P \\ T/P & \text{for } j = P \end{cases} \tag{4.5}$$

Finally, combining the results, the value of β_D using (3.35) is given by:

$$\beta_D \leq \sum \frac{\gamma_j^2}{\tilde{s}_{s(i)}(j)} \leq \left(\frac{T}{P}\right)^2 \cdot \left(\frac{\hat{\lambda}^2}{T \cdot h}\right) \ . \tag{4.6}$$

Then we can solve for the value T that satisfies the inequality:

$$T_D \geq \beta_D \cdot \left(\frac{P^2 \cdot h}{\hat{\lambda}^2}\right) \ , \tag{4.7}$$

where we replace T with T_D to indicate that it represents the time-to-detection for the sub-density containing the period P. From (4.7) we see that the time-to-detection depends on the period of the anomaly P, the width of the histogram bin h, and is inversely proportional to the combined traffic rate $\hat{\lambda}$. h appears in this inequality because we assume that the inter-arrival time P is exact and there is no jitter. In this case selecting h to be very wide allows more background inter-arrivals, those that fall in the same range $t \in \{j \cdot h, (j+1) \cdot h\}$ as P, to be in the same histogram bin as P. This increases the value of $s_{s(i)}(P)$ and reduces the value of $\beta_P = \frac{\gamma_P}{s_{s(i)}(P)}$. Thus, selecting a small value of h is desirable.

Rather than assuming all of the inter-arrival times between periodic packets are exactly equal to P, a more reasonable expectation is that the periodic traffic does experience some packet delay variation or jitter. Jitter in the timing of periodic packet arrivals causes the the inter-arrival times between periodic packets in the renewal density to vary around the exact value of P. The effect of jitter was described in detail in Section 4.2.3, where we described methods to estimate the width of the anomaly, denoted H_A based on the packet delay variation experienced as the anomaly traverses the Internet. A worst-case scenario would be that the jitter spreads the inter-arrival times, computed for the renewal density, generated from the periodic traffic uniformly across the entire width of the anomaly, H_A. In this case γ would be $\gamma_j = T/(P \cdot H_A)$ for all $j \in [-H_A/2, H_A/2]$.

If we assume the worst case scenario then the time-to-detection increases:

$$\beta_D \leq \sum_{j=P-H_A/2}^{P+H_A/2} \frac{\gamma_j^2}{\tilde{s}_{s(i)}(j)} \tag{4.8}$$

where we can replace γ_j by its value given the worst case scenario:

$$\beta_D \leq \sum_{j=P-H_A/2}^{P+H_A/2} \left(\frac{T}{H_A \cdot P}\right)^2 \cdot \left(\frac{\hat{\lambda}^2}{T \cdot h}\right) \tag{4.9}$$

Again solving for the value of T that satisfies this inequality (and renaming it T_D) we have:

$$T_D \geq \beta_D \cdot \left(\frac{H_A \cdot P^2 \cdot h}{\hat{\lambda}^2} \right) \tag{4.10}$$

Conveniently, the respective T_D's in the best case and worst case scenarios are related by a factor corresponding to the anomaly width, H_A.

Finally, the derivations of T_D given by (4.7) and (4.10) are determined for a specific period, P. In order to implement the detection system with the jumping window design described in Section 3.3.4 we need to determine a candidate time-to-detection for each sub-densities. Since the period of the actual anomalies is unknown *a priori* we select the largest time value (i.e., maximum period) that falls in the given sub-density. We use this maximum period to compute the value of T_D in a given sub-density. For instance, the maximum period of an anomaly in the first sub-density would be $P_1 = N_{bin} \cdot h$. Therefore, the base time-to-detection, $T_D(1)$, which is the time-to-detection in the first sub-density is found by substituting $P_1 = N_{bin} \cdot h$ into (4.7) or (4.10).

For the ith sub-density the period of the anomaly would be $P_i = i \cdot (N_{bin} \cdot h) = i \cdot P_1$. Substituting, $i \cdot P_1$ into either equation for time-to-detection it can be shown that $T_D(i) = i^2 \cdot T_D(1)$. Thus, the time-to-detection for the ith sub-density is i^2 times as long as the time-to-detection in the first sub-density.

4.3.1.2 Time-to-Detection for Subsampled Data

In the previous section the analysis assumed that subsampling was not applied to the measurement data. When subsampling is applied the expressions for the renewal density estimate, $\tilde{r}(j)$, and the smooth approximation, $\tilde{s}(j)$, must be modified to include the subsampling rate.

Combining the result from (3.24), giving the change in the renewal density due to thinning, with the expression in (4.3), the estimated renewal density for measurements from a Poisson process, when subsampling by a factor μ is:

$$\tilde{r}_\mu(j) = \frac{h}{\mu \cdot \hat{\lambda}} \tag{4.11}$$

Similar to the case in (4.4), we need to find the number of observed inter-arrival times equal to t, or more precisely the number of inter-arrivals that fall in the jth histogram bin. This is computed by multiplying the probability of such an inter-arrival time occurring, given by $\tilde{r}_\mu(t)$, by the total number of observed inter-arrival times. The total number of observed inter-arrival times is affected by subsampling as well because after subsampling the Poisson process has modified scale parameter $\mu \cdot \lambda$. Thus, combining (4.11) with the subsampled Poisson process the expected value of the estimate of the renewal density from (4.4) is modified to:

$$E[\tilde{r}_\mu(j)] = \frac{T}{\mu \cdot \hat{\lambda}} \cdot \frac{h}{\mu \cdot \hat{\lambda}} = \frac{T \cdot h}{\mu^2 \cdot \hat{\lambda}^2} \tag{4.12}$$

The value of γ_P was determined in the previous section to be $\gamma_P = T/P$. Following subsampling the number of periodic events sampled affects the value of γ_P. The probability that consecutive periodic events, which create an inter-arrival time equal to P, are both sampled is equal to $1/\mu^2$ because each of the periodic packets is sampled with probability $1/\mu$. Therefore after subsampling we have $\gamma_P = T/(P \cdot \mu^2)$.

Using the same analysis as for the non-subsampled case we can determine the value of T_D when subsampling is used to be:

$$\beta_D \leq \sum \frac{\gamma_m^2}{\tilde{s}_{s(i)}(m)} \leq \left(\frac{T}{P \cdot \mu^2}\right)^2 \cdot \left(\frac{\hat{\lambda}^2 \cdot \mu^2}{T \cdot h}\right) \tag{4.13}$$

Therefore the time-to-detection T_D is such that:

$$T_D \geq \beta_D \cdot \left(\frac{P^2 \cdot \mu^2 \cdot h}{\hat{\lambda}^2} \right) \tag{4.14}$$

Thus, we see that subsampling increases the time-to-detection by a factor of μ^2.

Similarly, for the worst case scenario then the time-to-detection increases by a factor of $H_A \cdot \mu^2$:

$$\beta_D \leq \sum_{m=P-H_A/2}^{P+H_A/2} \frac{\gamma_m^2}{\tilde{s}_{s(i)}(m)} \tag{4.15}$$

where we can replace γ_j by its value given the worst case scenario:

$$\beta_D \leq \sum_{m=P-H_A/2}^{P+H_A/2} \left(\frac{T}{H_A \cdot P \cdot \mu^2} \right)^2 \cdot \left(\frac{\hat{\lambda}^2 \cdot \mu^2}{T \cdot h} \right) \tag{4.16}$$

So that the time-to-detection is:

$$T_D \geq \beta_D \cdot \left(\frac{H_A \cdot \mu^2 \cdot P^2 \cdot h}{\hat{\lambda}^2} \right) \tag{4.17}$$

4.3.1.3 Theoretical Comparison

Using the analysis conducted in Section 4.3.1 we compare the theoretical time-to-detection for IA^{2}D using (4.7) and the bPDM algorithm using the average sample number (ASN) calculation given in [75]. The bPDM algorithm uses two separate statistical tests on the Internet traffic. The first test is based on the incoming packet rate, while a second test, based on packet size entropy, is used to prevent false positives. The ASN for the bPDM algorithm indicates the average number of samples before a detection decision can be made, where a sample is data from one uniform sampling interval. Therefore, the ASN multiplied by the sampling interval gives the average time-to-detection for the bPDM algorithm. While the ASN for the bPDM method can be computed for a more general class of Poisson processes, called the Generalized Poisson process [75], in this comparison we compute the ASN for the standard Poisson process. The ASN can only be calculated for the packet rate test, not the packet size entropy test. Thus, comparing our time-to-detection with the time-to-detection value for bPDM based on the ASN is fair since both both tests are based on packet timing, and by not using the packet size entropy test in the comparison neither method requires packet header information.

To compare the detection methods, multiple values of $\lambda = \{10, 13.3, 20, 40\}$ μseconds are selected, representing mean inter-arrival times for a Poisson process, as well as a range of periods, $P = \{200, 250, 333, 500, 1000, 1333, 2000\}$ in μseconds, to represent the periodic events. These inter-arrival times and anomaly periods are selected to simulate values commonly encountered in Internet traffic, and similar to those used in [75]. The probability of false alarm and probability of false negative are set to $P_{FA} = P_{FN} = 1\%$ for both methods. For our method we select $h = 1$ μsecond in order to match the time resolution for common Internet measurement systems, and select $N_{bin} = 50$. For the bPDM a uniform sampling interval of 1 millisecond is used, the value used in [75], except when this value produces less than one periodic event per sampling interval. In these cases, namely $P = \{1333, 2000\}$ μseconds the sampling interval is selected to be 1.5 and 2 milliseconds respectively. While this does bias the comparison in the favor of bPDM, since additional information is used in the selection of parameters, this modification is necessary otherwise the ASN calculation will not converge.

Figure 4.6 shows the theoretical time-to-detection curves for the two detection methods. Note that for $\lambda = \{10, 13.3\}$ and $P = \{1333, 2000\}$ the average sample number computation from [75] does not converge, so data points corresponding

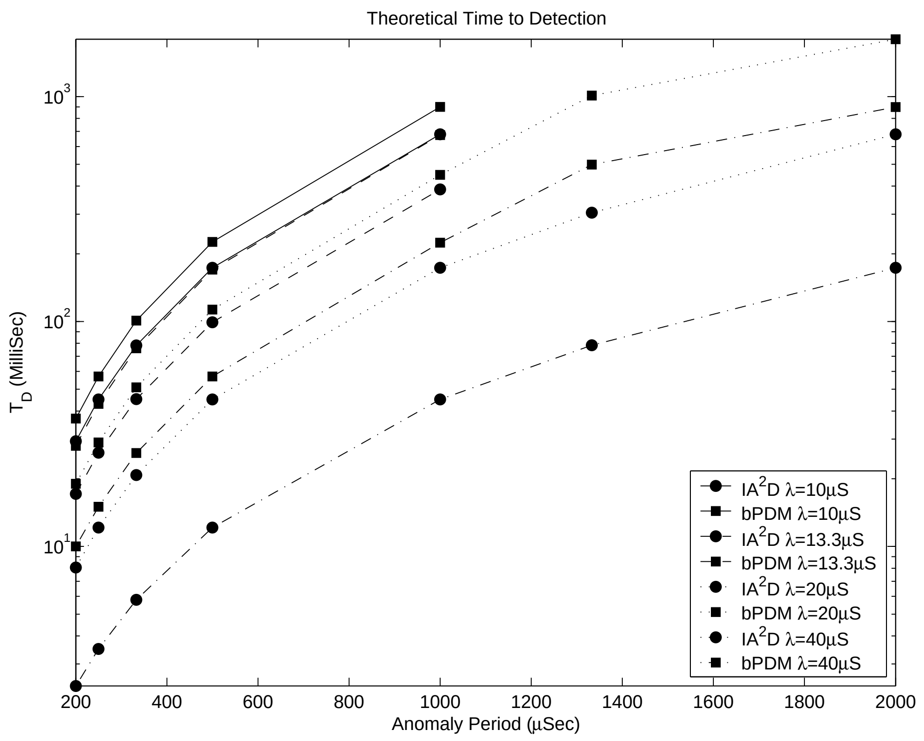

Figure 4.6: Comparing Theoretical T_D for IA^{2}D versus bPDM

to these values are omitted for both methods. In all scenarios time-to-detection is smaller for our method than for bPDM. In particular, when the period of the anomaly is on the order of bPDM's sampling interval, i.e., only one periodic event per sample, our method shows the largest gain over bPDM. Also, as mentioned previously, since bPDM required additional information about the anomaly being detected, our method has the advantage of still being able to detect the periodic anomaly when this additional information is unknown.

While IA^{2}D does show improvements over bPDM in time-to-detection, the main advantage of the method is it's ability to distinguish multi-period anomalies, which will be discussed in Section 4.3.3.

4.3.1.4 Experiments with Poisson Process

To support the theoretical analysis of the previous section here we conduct experiments that show IA^{2}D can detect multiple simultaneous periodic anomalies. For the experiments we generated random traffic with exponentially distributed inter-arrival times taking mean values of $\lambda = \{10, 13.3, 20, 25\}$ μseconds. We combined the random traffic with two periodic streams, labeled P_1 and P_2, taking values $P_1 = \{400, 800, 1333\}$ and $P_2 = \{1000, 2000\}$ μseconds. To prevent overlap the periodic stream starting times were given a random offset and no timing jitter was introduced.

Table 4.1 shows the average P_{FA} and P_{FN} for some P_1, P_2 combinations. The average P_{FA}, P_{FN} values were computed, for each P_1, P_2 combination, by running individual experiments with each value of λ and averaging the resulting P_{FA}, P_{FN} values. P_{FA} indicates the probability that the detector decides an anomaly is present in any of the sub-densities not containing one of the two periodic inter-arrival times or their harmonics, i.e., $k \cdot P_1, k \cdot P_2$. P_{FN} is the probability that the detector decides no anomaly is present in a sub-density containing one of the periodic inter-arrival times or their harmonics. The detection decisions were made in each sub-density after waiting $T_{D(i)}$ seconds, which was calculated using (4.7).

$T_{D(i)}$ was designed for $P_{FA} = P_{FN} = 1\%$, however, as shown in the table these exact values were not achieved. The cause for the deviation is primarily due to the variance in the histogram bin values, $\tilde{r}_s(j)$. The parameters used in calculating $T_{D(i)}$ are based on $E[\tilde{r}_s(j)]$, however, the actual $\tilde{r}_s(j)$ varies around the expected value, with variance $Var[\tilde{r}_s(j)]$. $Var[\tilde{r}_s(j)]$ impacts both P_{FN} by altering the value of γ, used in (4.7) and P_{FA} by making the assumption $\tilde{r}_s(j) = \tilde{s}_s(j)$, which we assumed true for random traffic, false. The effect of $Var[\tilde{r}_s(j)]$ is most prevalent when the

P_1 (μSec)	P_2 (μSec)	Avg $P_{FA}(\%)$	Avg $P_{FN}(\%)$
400	1000	6.44	1.39
800	1000	1.29	0.75
1333	2000	0.82	0.98

Table 4.1: P_{FA} and P_{FN} results averaged over values of $\lambda = \{10, 13.3, 20, 25\}$ μseconds

number of measurements per histogram bin is small, which occurs when T_D is short, e.g., in the lower subdensities or when the traffic rate is small. This is evident from the results, where P_{FN} for $P_1 = 400$ is larger than the other cases because T_D is shorter for $P_1 = 400$. To combat the effect of $Var[\tilde{r}_s(j)]$ it is possible to increase T_D, which would decrease P_{FN}, as indicated by the results in Table 4.1, where the experiments with P_1 equal to 800 and 1333 have lower $Avg\ P_{FN}$ than when $P_1 = 400$. Another possible way to mitigate the effect of $Var[\tilde{r}_s(j)]$ would be to change the width of the histogram bin h. Based on the expressions in (3.19) and (3.20) the variance in a histogram bin is inversely proportional to the width of the histogram bin, therefore, increasing the width h would decrease the variance in $\tilde{r}_s(j)$.

Finally, the very large value of P_{FA} for the experiments with $P_1 = 400$ and $P_2 = 1000$ is due to 400 and 1000 having similar harmonics, e.g., 800 and 1200 are harmonics of 400 and when combined with $P_2 = 1000$ the two periodic traffic streams produce inter-arrival times around 200 (depending on the initial alignment of the streams). Thus, the increase in P_{FA} is actually due to the periodic traffic, yet because the additional inter-arrival times are not harmonics of P_1 or P_2 the detections are counted as false positives. After removing the false positives created by the interaction of the two periodic traffic streams the P_{FA} values for the first experiment were more in-line with expectations, i.e., $P_{FA} = 1.1\%$.

4.3.2 Detection with Real Internet Traffic

In this section periodic event detection is examined using real Internet traffic. We use the results from the ideal Poisson analysis, T_D from (4.7), as a starting point in selecting the parameters of our system, and propose methods to modify the parameters in order to achieve reliable detection performance for the real Internet traffic case.

We begin by applying IA^{2}D in a detection simulation using real Internet traffic as the background traffic, with which we merge synthetic periodically-spaced traffic using the Stream Merger application [39]. Synthetic periodic traffic is used so that we can control the ratio of background to periodic traffic, and to have control over the amount of jitter in the periodic inter-arrival times. The background Internet traffic measurements we use in our simulations were collected at the LANDER Los Nettos trace collection system [20], and achieve an average data rate of around 300 Mbps and mean packet inter-arrival time of 13.18 μseconds. The periodic rates we use were $P = \{1.214, 1.735\}$ milliseconds. To simulate packet delay variation we add jitter to the periodic packet timestamps ranging uniformly over the interval of $[0, 12.5]$ μseconds for $P = 1.735$ milliseconds and jitter of $[0, 25]$ for $P = 1.214$. The jitter range only takes positive values because it simulates the packet being queued in a router, therefore, the packet is only delayed and never advanced. The additional systems parameter values are determined based on the methodology described in Section 3.3.5 and the selected values are: $H = 50$ μseconds, $h = 1$ μseconds, $Maxx =$ 2.5 milliseconds giving $Nsub = 50$. Note that the value of H is based on 2 to 4 times the width of the anomaly H_A. As described in Section 3.3.5.1, for the periodic event detection we assume the width is equal to the packet delay variation.

The resulting P_{FA} and P_{FN} values obtained using T_D from the ideal Poisson analysis in (4.7) are shown in Table 4.2. Based on these results we can determine

Subsampling Rate	Period mSec	$T_D(P)$ Sec	$P_{FA}(\%)$	$P_{FN}(\%)$
1	1.214	.565	6.54	85.61
1	1.735	1.1514	6.41	9.3

Table 4.2: P_{FA} and P_{FN} using T_D based on ideal calculations

that using T_D based on the ideal calculation will not work, primarily because of the presence of jitter in the periodic events. In particular, we note that jitter largely impacts the value of P_{FN} considering the large increase in false negatives when the jitter was increased for the $P = 1.214$ trial. To fix this problem we propose using the T_D calculation based on the worst case scenario described in Section 4.3.1. The worst case scenario calculation, given by Equation (4.10), is based on a uniform distribution of the periodic inter-arrival times, yet, according to the discussion in Section 4.2.3, the periodic inter-arrival times more closely follow a triangular distribution. However, we still propose using T_D based on (4.10) and consider it to be over dimensioning the system, i.e., we sacrifice detection time for detection accuracy. Using the proposed modification produces the results shown in Table 4.3.

Subsampling Rate	Period mSec	$T_D(P)$ Sec	$P_{FA}(\%)$	$P_{FN}(\%)$
1	1.214	28.25	10.29	0
1	1.735	28.75	9.16	0

Table 4.3: P_{FA} and P_{FN} using T_D based on worst case scenario

Therefore, by increasing T_D according to the worst case scenario we can reliably detect periodic anomalies, however, the value of P_{FA} is still to high for the system to be implemented in practice. The false positives in the detection system are created because as mentioned in Section 4.2.1 Internet traffic is not an ideal Poisson process so that the presence of "multi-level deterministic sequences" increases P_{FA}. To reduce the number of false positives we propose the use of back-to-back packet filtering as described in Section 4.2.2. Filtering out the back-to-back measurements in our experimental data sets produces the equivalent subsampling rate of approximately 1.2. Using this value as our subsampling rate we determine T_D based on the worst case scenario calculation for subsampled data, given by Equation (4.17). Note that the equivalent subsampling rate due to back-to-back packet filtering is data dependent. Therefore, when back-to-back packet filtering is used in practice the equivalent subsampling rate will have to be computed from the measurements, i.e., taking a time average of the number of packets per measurement $C(k)$. The equivalent subsampling rate can then be used for μ in (4.17). Using the new T_D and applying IA^{2}D to the filtered measurements the system achieves the results shown in Table 4.4.

Subsampling Rate	Period mSec	$T_D(P)$ Sec	$P_{FA}(\%)$	$P_{FN}(\%)$
B2B	1.214	40.68	1.33	0
1.2(*)	1.214	40.68	6.35	0
6(*)	1.214	1017	1.13	0
B2B	1.735	41.4	1.38	0
1.2(*)	1.735	41.4	6.29	0
6(*)	1.735	1035	1.16	0

Table 4.4: P_{FA} and P_{FN} using back-to-back (B2B) packet filtering and worst case T_D

For comparison we include in Table 4.4 the results using standard subsampling, with the (*) at the same rate achieved by the back-to-back filtering. While standard subsampling does improve P_{FA} compared to no subsampling, the performance of back-to-back packet filtering is significantly better and nearly in line with the design goal of $P_{FA} = 1\%$. Finally, we also include the result based on subsampling by a factor of 6, which shows that if one desires even better performance than that achieved by back-to-back packet filtering, it is possible to achieve it via subsampling, however, this comes at a cost of a significant increase in T_D.

Finally, similar to the results shown for a Poisson process we now show some results

μ	P_1 (μSec)	P_2 (μSec)	Avg P_{FA}(%)	Avg P_{FN}(%)
1	800	1333	10.91	0
1.2	800	1333	7.86	0
B2B	800	1333	1.86	0
1	1333	2000	12.31	0
1.2	1333	2000	9.91	0
B2B	1333	2000	3.18	0

Table 4.5: P_{FA} and P_{FN} results for $\mu = \{1, 1.2, B2B\}$

of applying IA^{2}D to real Internet traffic combined with multiple periodic events. For this simulation we selected two period combinations, $\{P_1 = 800, P_2 = 1333\}$ μseconds and $\{P_1 = 1333, P_2 = 2000\}$ μseconds, which are identical to the two of the values used in Section 4.3.1.4. T_D was selected using the worst case scenario value in (4.17) and we show results for subsampling rates $\mu = \{1, 1.2\}$ and back-to-back packet filtering. The results for multi-period detection are shown in Table 4.5. Note that the value of P_{FN} is computed at both the fundamental periods, P_1 and P_2, and any of their harmonics. Also, the value of P_{FA} is determined from any subdensity not containing a harmonic of one of the periodic components. As discussed in Section 4.3.1.4 when multiple periodic events exist they create additional periodicities related to the sum and difference of their periods. Because of this the values of P_{FA} are marginally higher than was the case in the single periodic event case. As was done in Section 4.3.1.4 when we remove the false positives that are due to the interaction of the two periodic traffic streams the value of P_{FA} decreases to similar values as were achieved in the single periodic stream experiment, i.e., approximately 1.3 (Table 4.4).

4.3.3 Quasi-Periodic Anomaly Detection

In the previous section we considered anomalies which were generated at a fixed period P. Here we consider a slight variation that could potentially occur if the attack was launched by a smart adversary, i.e., someone who knew the operation of the detection mechanism.

We define a quasi-periodic anomaly as one that has a base period, P, yet the actual events occur at random multiples of this period, i.e., the time difference between consecutive events in the attack is $\alpha \cdot P$ for some randomly generated value α. For simplicity we only consider the case where α is geometrically distributed with mean value $1/q$. We consider this case because from the attackers viewpoint it is easy to implement, yet because the geometric distribution creates a uniform distribution of inter-arrival measurements in the renewal density (as described with respect to our subsampling optimization method in Section 2.3) this type of quasi-periodic structure is the most difficult for our method to detect. $\alpha \sim Geo(q)$ is obtained if every P seconds the attacker decides to output an attack packet with probability q or decides to not output an attack packet with probability $1-q$. Given this scenario the probability of having an inter-arrival equal to P is only q, which clearly will create problems in our system since this was not considered in the selection of T_D. However, given that our system performs detection on sub-densities we can instead detect a harmonic, or a series of harmonics of the base period without having to redesign the selection of T_D.

The analysis presented here works for any value P and any value of q. In other words the attack could select a very short P and make q very small, or the attack could choose both P and q to be large. Both of these choices would produce a similar average attack packet rate, yet the structure of the attack would be different. However, using the methodology presented in the following analysis works for any value for P and q.

In order to show that our system can detect quasi-periodic attacks using harmonics *without having to redesign the selection of* T_D we present the following analysis. First, consider the value of γ at harmonics of P. For the *non-random* periodic attack case, let $N(t)$ be the number of inter-arrivals, in t seconds equal to P due to the periodic attack. The value of γ (3.32) at P, which we defined in Section 3.3.2 as the component of the renewal density estimate due to the anomaly, is given by $N(t)$. For the mth harmonic of P the number of inter-arrivals equal to $m \cdot P$ is equal to $N(t) - m$ or approximately $N(t)$ if $N(t)$ is large. Therefore, the value of γ at harmonics of P is approximately equal to γ at P itself, i.e., $\gamma_{mP} \approx \gamma_P$. Because the value of γ remains the same at harmonics of the base period the time-to-detection should be the same in the sub-density containing the base period P as it is for the sub-density containing a harmonic. However, the actual time-to-detection, calculated using (4.7) or (4.10), for the sub-density containing the harmonic is equal to $T_D(mP) = m^2 \cdot T_D(P)$, i.e., the T_D at the mth harmonic is m^2 times T_D at the base period. In other words, the value of T_D at a the mth harmonic is m^2 times what it actually needs to be, i.e., it is over-dimensioned.

Next we consider the case of the quasi-periodic attack. If $N(t)$ is as defined above, and if α is geometrically distributed, then the probability of an inter-arrival equal to P is equal to q. Therefore, the number of inter-arrivals, in the renewal density, equal to P in the quasi-periodic case is $q \cdot N(t)$. Given this the value of γ_P when the probability of an inter-arrival equal to P is equal to q is equal to $\gamma_P(q) \approx q \cdot \gamma_P$. This decrease in γ requires a corresponding increase in T_D equal to $T_D(P_q) = T_D(P)/q^2$. In other words, the decrease in γ would require that we over-dimension by a factor of $1/q^2$, if the value of q was know *a priori* (which it is not).

In order to determine the probability of having an inter-arrival equal to a specific harmonic, i.e., $m \cdot P$ we can reuse the inter-arrivals retained function $I(n)$, given in (2.4) (Section 2.3.2) that was used to determine the inter-arrivals retained at a specific order based on the method of interrupt coalescence used. We can use $I(n)$ because essentially what the attacker is doing by randomly selecting a multiplier α for the base period is subsampling the periodic events, so the calculation using $I(n)$ is the same for both cases. Similar to the result in Section 2.3.2 by using a geometric distribution for α, the inter-arrivals retained function is constant for all n and equal to q. Thus, the number of inter-arrivals equal to *any* harmonic of P will be $q \cdot N(t)$. Finally, as was the case for the base period the value of γ at the mth harmonic is approximately equal to $\gamma_{mP}(q) \approx \gamma_P(q) \approx q \cdot \gamma_P$. Similarly, the corresponding increase in T_D is $T_D(mP_q) = T_D(mP)/q^2$ However, as was shown previously the value of $T_D(mP)$ is actually over dimensioned by a factor of m^2 automatically for harmonics of the base period. Thus, at the $m = 1/q$th harmonic, the necessary increase in T_D due to the quasi-periodic behavior of the anomaly is offset by the automatic over dimensioning due to the harmonics of the base period. Note that $1/q$ represents the subsampling rate generated by selecting $\alpha \sim Geo(q)$, i.e., $1/q = E[\alpha]$. This result indicates that if an attacker generates a periodic attack using randomly selected multiples of the base period, i.e. $\alpha \cdot P$, where $\alpha \sim Geo(q)$, then we can detect the anomaly using in the $m = E[\alpha]$th harmonic without modification to the existing T_D calculations. In other words, the over-dimensioning of T_D at the harmonics of P correspond to the necessary increase in T_D because of the quasi-periodicity of the attack. Further, this over-dimensioning is done automatically because of how the detection system operates and does not require any knowledge about the structure of the quasi-periodic attack.

While we only considered here the case where $\alpha \sim Geo(q)$ the result that we can detect a quasi-periodic anomaly using one, or a series of its harmonics without modification to the T_D value applies to more general distributions of α. This results holds for a more general distribution of α because the inter-arrivals retained function, $I(n)$, converges to the reciprocal of the the subsampling rate, i.e., $1/E[\alpha]$, as n gets large for most distributions of α. A suitable condition on the distribution of α is that all values in the support of α have a non-negligible probability mass, i.e., $p(\alpha) > 0 \ \forall \alpha \in [0, \psi]$ for some value ψ. This condition is suitable because $I(n)$ for large n is the sum of many terms, where each term is the product of multiple probability mass weights, i.e., each term is $p(\alpha_1) \cdot p(\alpha_2) \cdots p(\alpha_j)$. Even if the $p(\alpha_j)$ are given different weights, as they are for different distributions of α, when the product of many $p(\alpha_j)$s is taken, as is the case for large n, the result is similar. However, this condition does not hold when some of the $p(\alpha_j)$ are zero because the product of multiple terms would then be zero, thus our condition that $p(\alpha) > 0$ over some support is suitable. Therefore, each term in the sum for $I(n)$ is similar regardless of the distribution of α, and the sum of these similar terms gives a similar value for $I(n)$ regardless of the distribution of α. The convergence of the $I(n)$ regardless of distribution of α can be seen in Figure 2.11, where as n increases the KL divergence for the different subsampling methods converges. The value of the KL divergence converges because the inter-arrivals retained for each subsampling method converges as n increases. Further, we note that it was observed that the value of $I(n)$ begins to converge quickly for $n > \psi$, i.e., for n larger than the support of α. Therefore, we can use the result obtained in this section, that we can detect quasi-periodic anomalies using the mth harmonic of the base period P without modification of T_D, for a general distribution of α assuming that $m > \psi$. The condition $m > \psi$ will be to strong for most distributions of α, however, it gives a general sufficient condition.

To verify this result we conducted an experiment by generating a quasi-periodic anomaly with base period $P = 1.333$ milliseconds, and using the value of $q = .33$ indicating that we should be able to detect the anomaly in the 3rd harmonic using the standard calculation of T_D. The results of the experiment at various subsampling rates are shown in Table 4.6. Notice from the results how the value of P_{FN} is initially very high at the base period, and continually decreases until the third harmonic where

				P_{FN}		
μ	P (μSec)	E[k]	Avg P_{FA}(%)	P	$2 \cdot P$	$3 \cdot P$
1	1333	3	16.63	72.5	1.17	0
1.2	1333	3	13.6	76.19	13.03	0
B2B	1333	3	6.37	94.56	14.61	0
3	1333	3	9.99	95.4	0.31	0.31
3 (B2B)	1333	3	1.40	96.31	26.34	0.98

Table 4.6: P_{FA} and P_{FN} results detecting quasi-periodic attack

the attack is detected with near perfect accuracy.

One important thing to note about this experiment is that because we have to perform detection looking for harmonics of the base period we had to select $Maxx$ to be larger as well. This led to increased P_{FA} because at larger inter-arrival times the deviation of the Internet traffic from the Poisson process becomes more apparent. We can compensate for this deviation however, by increasing the rate of subsampling, and combining subsampling with using back-to-back packet filtering. The best result was obtained using back-to-back packet filtering combined with additional subsampling such that the combined subsampling rate was 3. From the table we see that this combination produces a reasonable P_{FA} of around 1.4% with $P_{FN} < 1\%$.

Thus, we have shown that our IA^{2}D detection system can be used even when the attacker uses counter measures to prevent their attack from being detected.

4.4 TCP-Targeted Anomaly Detection

The second detection application we propose to use IA^{2}D for is to detect low-rate TCP-targeted DoS, or shrew attacks for short. A shrew attack is very different from the periodic anomaly case, in that it consists of periodically spaced bursts of packets. The spacing of packets inside a burst is entirely random. Therefore, our system aims at detecting the periodic characteristic of the bursts and not the packets themselves.

Figure 4.7 shows an example of how a shrew attack operates. The figure shows the

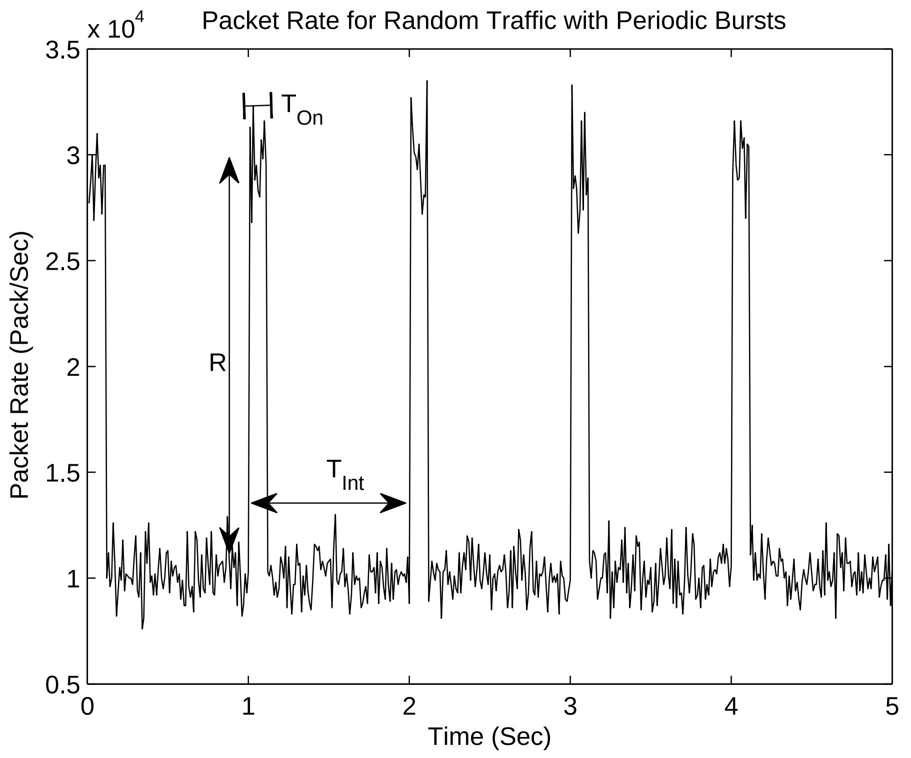

Figure 4.7: Instantaneous Packet Rate with Periodic Bursts due to Shrew Attack

instantaneous packet rate measured on the link where the attack is being detected. The large, and short-lived, increase in the packet rate is the signature of the shrew attack. The shrew attack has three parameters: R is the packet rate of the attack when it is active, T_{on} is the duration of the attack burst, and T_{Int} is the interval between packet bursts. We do not assume that the attack packets within a burst are periodic, instead we assume that the packets are uniformly distributed within the burst. Within the renewal density estimate, using the above assumptions, the shrew attack manifests itself as triangular shaped distributions with width $2 \cdot T_{on}$ located around multiples of T_{Int} as shown in Figure 4.8. Note that the triangular distribution of the shrew attack inter-arrival times is due to the uniform distribution of the packets within a burst. For a different distribution of the packets the distribution of the inter-arrival times will be different as well. Different distributions of inter-arrival times in the renewal density can potentially increase the time-to-detection from the values we compute using our ideal analysis. However, by over-dimensioning the value of T_D using a "worst-case scenario", as was done for the periodic event detection, our method should be able to detect anomalies with a variety of inter-arrival distributions. Note that the examples in Figures 4.7 and 4.8 were generated using ideal randomly

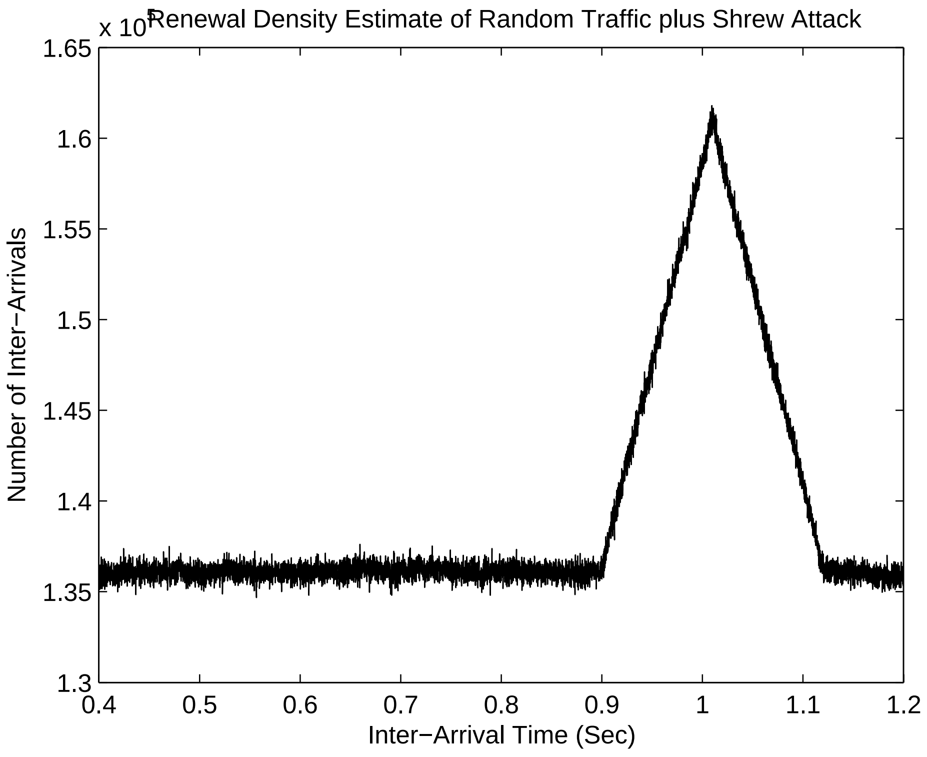

Figure 4.8: Example of Renewal Density Estimate with Shrew Attack Triangular Peaks

generated measurements. In real traffic the peaks in the renewal density estimate are not as clearly defined.

We begin, in Section 4.4.1, with an analysis of the system using a Poisson process to model the Internet traffic. This is combined with periodic bursts of higher rate uniformly distributed traffic to model the shrew attack. As before we use this analysis to derive an expression for the time-to-detection T_D, which in the case of a shrew attack depends on the attack parameters, R, T_{on}. In practice we cannot assume that these parameters will be known *a priori*, however, the user implementing the system can determine, based on the capabilities of their network what would be minimal values of R and T_{On} which would disrupt their network operation. These values could then be used to determine base parameters for the system, which would automatically detect attacks with higher R or longer T_{On} as well.

The structure of the shrew attack is quite different from the periodic events detected in Section 4.3, and presents some unique challenges when we attempt to perform detection using real Internet traffic. These challenges, which include the longer time scale between periodic events, i.e., bursts, are discussed in Section 4.4.2 along with some techniques to ameliorate these problems.

4.4.1 Ideal Analysis

In this section we present an analysis of detection based on using a Poisson process to model the Internet traffic. This analysis leads to ideal and worst case expressions for the time-to-detection which in the case of a shrew attack depends on the parameters of the attack, i.e., $T_D(R, T_{on}, T_{Int})$.

Much of the analysis is the same as that of Section 4.3.1, so we will refer to many of equations derived there. However, unlike the periodic event detection scenario where we were able to use the smooth approximation for renewal process (Section 3.3.3.1), here we must rely on the smooth approximation technique for non-renewal processes (Section 3.3.3.2), i.e., the trimmed mean filtering approach. We must use the trimmed mean method because due to the structure of the shrew attack we cannot guarantee that the first order inter-arrival distribution, $\tilde{f}_1(j)$ will be anomaly free, in fact during the high-rate bursts back-to-back attack packets are common. Also, because the shrew attack appears in the renewal density as a triangular distribution, this requires that we increase the width of trimmed mean filter. Previously the width of the filter, $2 \cdot J$, was selected to be equal to width of the sub-density, $h \cdot N_{bin}$, yet the width of the sub-density is selected to be 2 to 4 times the width of the anomaly H_A. Keeping the width of the trimmed mean filter to be 2 to 4 times the width of the anomaly often times does not completely remove the presence of the shrew attack from the smooth approximation. Therefore, we recommend selecting the width of the filter to be $2 \cdot h \cdot N_{bin}$ to sufficiently remove the presence of the shrew attack from the smooth approximation. As mentioned in Section 3.3.5.1 the width of the anomaly for the shrew attack depends on the value of T_{On}, and because we do not know the actual value of T_{On} we select H_A to be equal to the maximum value of T_{On} that was shown in [10,46] to be the used in practical TCP-targeted DoS attacks which was 100 milliseconds.

The following ideal analysis assumes that we are able to remove the triangular peak in the renewal density due to the anomaly using the trimmed mean filtering approach. In our work we use the trimmed mean filtering method to generate the smooth approximation primarily because that is what was used for periodic event detection. Other methods could be used that are better suited to remove the presence of the triangular peak in the renewal density.

Consider detection in subdensity i, the value of the smooth approximation at index m in the subdensity, $\tilde{s}_{s(i)}(m)$, is given by Equation (4.4) where in this case $\hat{\lambda}$

represents the average inter-arrival time of the combined traffic (Internet traffic plus shrew traffic). The value of $\hat{\lambda}$ is given by:

$$\hat{\lambda} = \lambda + \frac{T_{Int}}{R \cdot T_{On}} \tag{4.18}$$

Next we determine the value of $\gamma(m)$, which represents the difference between the renewal density estimate, $\tilde{r}_{s(i)}(m)$, and the smooth approximation, $\tilde{s}_{s(i)}(m)$, for histogram bins where the presence of the attack occurs. Since the shrew attack manifests itself differently in the renewal density estimate, as shown in Figure 4.8, determining γ_m is not as straight-forward as it was for the periodic event case. Since γ_m is the difference between the smooth approximation and the renewal density estimate its value is equal to the value of the triangular distribution shown in the figure, and determining the value of γ_m is done as follows. The number of packets per burst in a shrew attack is given by, $R \cdot T_{On}/\mu$, where μ is the subsampling factor ($\mu = 1$ if subsampling is not used). The total number of inter-arrival times generated by a pair of packet bursts is approximately, $(R \cdot T_{On}/\mu)^2$. The number of bursts per second is given by T/T_{Int}, and combining with the number of inter-arrivals per burst gives the number of inter-arrivals per second due to the attack:

$$\frac{T \cdot (R \cdot T_{On})^2}{\mu^2 \cdot T_{Int}} \tag{4.19}$$

This total number of inter-arrival times is spread across the triangle distribution in the renewal density estimate, centered at multiples of T_{Int}, as depicted in Figure 4.8. The width of the triangle distribution depends on the width of the histogram bins, and is equal to $2 \cdot T_{On}/h$. The maximum value of the triangle distribution, which occurs at the inter-arrival time equal to T_{Int}, is given by:

$$\gamma_{Tint}(T) = \frac{T \cdot h \cdot (R \cdot T_{On})^2}{\mu^2 \cdot T_{Int} \cdot T_{On}} \tag{4.20}$$

The value of γ at index $T_{Int} \pm k$ where $k \in [-T_{On}, T_{On}]$ is given by the following expression:

$$\gamma_{Tint+k}(T) = \left(1 - \frac{|k|}{T_{On}}\right) \frac{T \cdot h \cdot (R \cdot T_{On})^2}{\mu^2 \cdot T_{Int} \cdot T_{On}} \tag{4.21}$$

From (4.7) we can determine the time-to-detection for the subdensity containing T_{Int} by,

$$\beta_D \leq \sum_{k=-T_{On}}^{-T_{On}} \frac{\gamma_{Tint+k}(T)^2}{\tilde{s}_{s(i)}(T_{Int} + k)} \tag{4.22}$$

substituting the values of γ from (4.20) and (4.21):

$$\begin{aligned} \beta_D &\leq \sum_{k=-T_{On}}^{-T_{On}} \left[\left(1 - \frac{|k|}{T_{On}}\right) \frac{T \cdot h \cdot (R \cdot T_{On})^2}{\mu^2 \cdot T_{Int} \cdot T_{On}}\right]^2 \cdot \left(\frac{\mu^2 \cdot \hat{\lambda}^2}{T \cdot h}\right) \\ \beta_D &\leq \frac{T \cdot h \cdot \hat{\lambda}^2 \cdot R^4 \cdot T_{On}^3}{\mu^2 \cdot T_{Int}^2} \cdot \sum_{k=-T_{On}}^{-T_{On}} \left(1 - \frac{|k|}{T_{On}}\right)^2 \end{aligned} \tag{4.23}$$

Replacing T with the time-to-detection T_D and solving for T_D:

$$T_D \geq T_{Int}^2 \cdot \frac{\mu^2 \cdot \beta_D}{h \cdot \hat{\lambda}^2 \cdot R^4 \cdot T_{On}^3 \cdot \sum_{k=-T_{On}}^{-T_{On}} \left(1 - \frac{|k|}{T_{On}}\right)^2} \tag{4.24}$$

Similar to the periodic event detection analysis when subsampling is applied T_D increases by a factor of μ^2.

The implementation the jumping window mechanism in IA²D using T_D from (4.24) is done in the same way as was done for the periodic event detection system except that in (4.24) T_{Int} replaces P. $T_D(1)$, the time-to-detection in the first subdensity, is determined from (4.24) by using $T_{Int}(1) = N_{bin} \cdot h$. As before time-to-detection in the ith subdensity is related to $T_D(1)$ by $T_D(i) = i^2 \cdot T_D(1)$, which makes determining jumping window lengths for the implementation of IA^{2}D convenient.

4.4.2 Actual Traffic

Using IA^{2}D to detect low-rate TCP-targeted DoS attacks in practice has many more complications that were encountered for the periodic event detection. The first issue is that the values of R and T_{On} are not known *a priori*. As mentioned in Section 3.3.5.1 we select the width of the subdensities based on the maximum value of T_{On} that is commonly encountered in real life shrew attacks, which is equal to 100 milliseconds according to [10,46]. Then the value of R can be determined by the user based on the minimum attack rate per burst that would compromise their network, where attack rate per burst is given by $R \cdot T_{On}$. By using the minimum value of attack rate per burst, and the minimum value of the burst interval the system is over dimensioned and therefore will detect attacks at higher rates as well.

In the following experiments instead of using the minimal attack rate we use the actual values of R and T_{On}, which are known because we generate the shrew attacks. This is done such that we can test the expression derived for time-to-detection in (4.24). For this experiment we use background traffic from the Abilene-III Internet trace dataset from the WAND research group at the University of Waikato [1], which was also used in simulations by another TCP-targeted DoS attack detection system in [10]. The average traffic rate for this dataset was approximately $82,000$ packets per second. For our simulations we generated two shrew attacks with the following parameters:

1. $R = 31250$, $T_{On} = 50$ milliseconds and $T_{Int} = .95$ seconds

2. $R = 31250$, $T_{On} = 30$ milliseconds and $T_{Int} = 1.9$ seconds

These values we selected to be similar to the values used in [10]. Just as was done when generating traces for the periodic event simulation the real Internet traffic was merged with the simulated shrew attacks using the Stream Merger application [39].

The additional systems parameter values are determined based on the methodology described in Section 3.3.5. As mentioned previously the width of the anomaly H_A is selected to be 100 milliseconds, and the subdensity width was selected to be twice this value, i.e.,$H = .2$ seconds. We used $N_{bin} = \{50, 25, 10\}$ corresponding to values of $h = \{4, 8, 20\}$ milliseconds respectively. The different choices for N_{bin}/h did not show a significant difference, therefore, the results from each are averaged together in the tables below. We use a trimmed mean filter with a width of $2 \cdot H = .4$ seconds in order to remove the majority of the triangular distribution.

In our simulations we apply subsampling to the datasets for two reasons. The first reason is practical, in that for detecting shrew attacks the order of the inter-arrivals computed to estimate the renewal density is larger than for periodic event detection. For periodic event detection the maximum inter-arrival time $Maxx$ was on the order of milliseconds, however, to detect shrew attacks requires $Maxx > T_{Int}$, which means $Maxx$ must be of the order of seconds for our simulations. With $82,000$ packets per second, on average, this requires computing inter-arrivals of order at least $300,000$ (twice the value of $1.9 \cdot 82,000$), which is approximately 1000 times larger than the inter-arrival order required for periodic event detection. The second reason, discussed in Section 4.2.1, is that as the inter-arrival order and times increase the deviation of the Internet traffic from the Poisson process assumption become exaggerated and higher subsampling rates are required to minimize this deviation. The subsampling rates applied in the following simulations were $\mu = \{10, 25, 50, 100\}$.

The first set of simulations run on the trace datasets used the value of T_D determined by (4.24) and the values of P_{FA} and P_{FN} for the varying subsampling rates are shown in Table 4.7

Subsampling Rate	T_{On} mSec	T_{Int} Sec	$T_D(T_{Int})$ Sec	$P_{FA}(\%)$	$P_{FN}(\%)$
10	50	.95	.3	57.65	3.89
25	50	.95	1.82	25.36	4.98
50	50	.95	7.26	9.91	7.65
100	50	.95	29.04	2.43	8.23
10	30	1.9	6.81	86.17	4.68
25	30	1.9	42.54	31.51	5.68
50	30	1.9	170.14	11.18	8.47
100	30	1.9	680.54	2.80	9.32

Table 4.7: P_{FA} and P_{FN} using T_D based on ideal calculation

Two things are evident from this experiment. First the value of T_D based on the ideal calculation is insufficient to produce reliable detection results, and second higher subsampling rates are required to maintain values of P_{FA} for a usable system.

In order to reduce the value of P_{FN} we propose the following modification to the calculation of T_D: remove the triangle distribution assumption and instead use just the maximum value of γ which occurs when $k = 0$ or at T_{Int} in (4.21). Using this assumption and including the subsampling rate parameter μ the new value of T_D is given by:

$$T_D \geq T_{Int}^2 \cdot \frac{\mu^2 \cdot \beta_D}{h \cdot \hat{\lambda}^2 \cdot R^4 \cdot T_{On}^2} \tag{4.25}$$

Using T_D determined by (4.25), the previous simulations were repeated and the results are shown in Table 4.8. From these results we see that the increased time-to-detection is sufficient to provide reliable detection while only increasing P_{FA} negligibly. Note that we do not show the results for subsampling rates 10 and 25 as they generate large values of P_{FA}, which makes these rates unusable in a practical

Subsampling Rate	T_{On} mSec	T_{Int} Sec	$T_D(T_{Int})$ Sec	$P_{FA}(\%)$	$P_{FN}(\%)$
50	50	.95	18.15	10.25	0
100	50	.95	72.60	2.97	0
50	30	1.9	255.20	10.43	0
100	30	1.9	1020.80	3.18	0

Table 4.8: P_{FA} and P_{FN} using T_D based on the modified calculation

detection system. If it is necessary to use the lower subsampling rates, for faster time-to-detection, a two-stage detection method could be implemented. IA^{2}D could be used as the first stage, and then a second stage to filter the detection results could be used to reduce false positives. The filtering could be done based on properties of the anomaly, i.e., the triangular distribution shown in Figure 4.8, which is unique to the anomaly and would likely not appear due to Internet traffic. The triangular distribution of the shrew attack means that the difference $\tilde{r}(j) - \tilde{s}(j)$ is mainly a positive value in histogram bins containing the shrew attack. In other histogram bins the same difference, due to Internet traffic alone, would range both positive and negative. A potential filtering technique would be to examine the sub-densities where an anomaly was detected and determine if the detection was due to differences, $\tilde{r}(j) - \tilde{s}(j)$, that were mainly positive, indicating a shrew attack, or differences that were randomly distributed positive and negative, indicating a false alarm.

Therefore, by modifying the ideal T_D calculation and using the over dimensioned value determined in (4.25) our detection system is capable of reliably detecting shrew attacks when large subsampling rates are selected. While the larger subsampling rate significantly increases T_D its use is necessary to reduce P_{FA}, as shown in the results, as well as beneficial because it reduces computation time and storage cost for the measurements recorded for detection. Thus we have shown that IA^{2}D can potentially be used in the detection of another kind of periodic anomaly commonly found in Internet traffic.

4.4.3 Comparison to Related Work

Most methods to detect shrew attacks require separating Internet traffic into flows, which are collections of packets with identical source/destination IP addresses; for example, representing a connection between a client and server in single TCP session. One method using flow separation, described in [10, 11], filters out shrew attacks using a sequential hypothesis test that decides if a flow is a shrew attack by examining the spectral characteristics, which are different for shrew and standard TCP flows. Results reported in [10] indicate detection rates for shrew attacks of 99.7%, while falsely reporting regular flows as attacks $< 3\%$ of the time. Therefore, our results for shrew detection are comparable to those of [10]. However, our detection has some advantages. The clear advantage is that our method can detect shrew attacks, in aggregate traffic, with the same accuracy as the method in [10], which requires flow separation. This ability to work with aggregate traffic comes at a cost of increased time-to-detection. As shown in the previous section due to the high rate of sub-sampling required in practice the time-to-detection of our system for shrew attacks is longer than it was for periodic event detection. The authors of [10], however, do not report the performance of their system in terms of time-to-detection, nor do any of the other works on shrew detection, therefore we cannot speculate on how our time-to-detection relates to that of comparable systems.

The other advantage our method has is that by working with aggregate traffic we can perform detection even when IP addresses are encrypted or spoofed, which makes flow separation inaccurate or impossible, impacting the performance of the detection method from [10]. In fact, the authors in [10] acknowledge that source IP address are often spoofed in attack packets, yet declare that this is not a problem for their system; this is a questionable assertion if the attack uses smart adversary techniques such as random generation of source IP addresses.

4.5 Practical System Implementation

The experiments conducted in this chapter were designed and performed to verify that our analysis, with modification for real Internet traffic, could be used to calculate the time-to-detection for desired values of P_{FA} and P_{FN}. In a practical system implementation, however, the most crucial factor is the probability of false alarm since an alarm being asserted requires user intervention to verify that indeed an anomaly is present. A false alarm rate of 1% is too high if the system is outputting detection decisions from many sub-densities, e.g., 100, on the order of once per second, i.e., one false alarm per second.

Being that our system is designed to detect very low-rate anomalies, it is more realistic that the user would be willing to sacrifice time-to-detection in order to lower the probability of a false alarm. In other words, detecting a very low-rate anomaly in seconds is not as crucial as detecting it in minutes, while having the user only have to check on a false alarm once per 30 minutes rather than once per 30 seconds is much more important.

We can lower the false positive rate in a number of different ways. The first way to lower the false positive rate is to simply set the value of P_{FA} to be very low, e.g., $P_{FA} = 0.001\%$. Changing P_{FA} affects the value of the threshold χ^2_D (see (3.34)) which impacts the value of β_D (see (4.1)) and finally increases the time-to-detection, T_D. In Figure 4.9 we show the percentage increase in β_D plotted for $P_{FA} = \{.01, .005, .001, .0005, .0001, .00001\}$ on a logarithmic scale. The values of β_D are normalized by the value of β_D when $P_{FA} = .01$, and then multiplied by 100 to express them as percentages. We use β_D when $P_{FA} = 0.01$ as the baseline value since this is the value we used for the experiments in this chapter. Since T_D is directly proportional to the value of β_D (for instance in (4.7) or (4.24)) the increase in β_D also indicates the increase in T_D.

In Figure 4.9, we also plot the β_D curves for multiple values of N_{bin}. When the value of N_{bin} is increased the percent change in β_D due to lowering P_{FA} is actually lessened. Note that the actual value of β_D is larger for larger N_{bin}, but the percent increase from the baseline β_D in each trial is lower. Therefore, even when large values of N_{bin} are used the percentage increase in β_D (and subsequently T_D) are small.

We see from Figure 4.9 that lowering the false positive rate does not produce a significant increase in the time to detection. For example, lowering P_{FA} from 0.01 to 0.001, a factor of 10, only corresponds to an increase in T_D of around 160% or 1.6 times. Using the result in Table 4.4, where T_D was around 40 seconds when using back-to-back packet filter, the new T_D for $P_{FA} = 0.001$ would only be around 65 seconds.

To verify the results shown in Figure 4.9 we ran the same experiment from Section 4.3.2 using back-to-back packet filtering and T_D based on the worst case scenario, however, we changed the desired P_{FA} from 0.01 to 0.0001. This increased the T_D value (from Table 4.4) from approximately 41 seconds to 76.67 seconds, an increase of 187%. After running the simulation for both periodic anomalies considered in Section 4.3.2 the average P_{FA} obtained was 0.0007, which is higher than our desired value.

To look at the cause of this result we consulted the data, and discovered that multiple false positives in a particular sub-density typically occurred in consecutive detection decision intervals. In other words, the same sub-density was reporting a false positive multiple times in a row. This discovery makes intuitive sense given that detection is performed using a jumping window. Therefore, if there is some spurious inter-arrival data that causes the false positive it will actually appear in all detection decisions until the jumping window has shifted beyond the spurious data. By removing the multiple false positive values from our detection results, and considering only the initial detection of a false positive in each sub-density the value of P_{FA} dropped below 0.0001 to approximately 0.00008. Removing the multiple false positive values is not something that was considered in the previous experiments, in Sections 4.3.2 and 4.4.2.

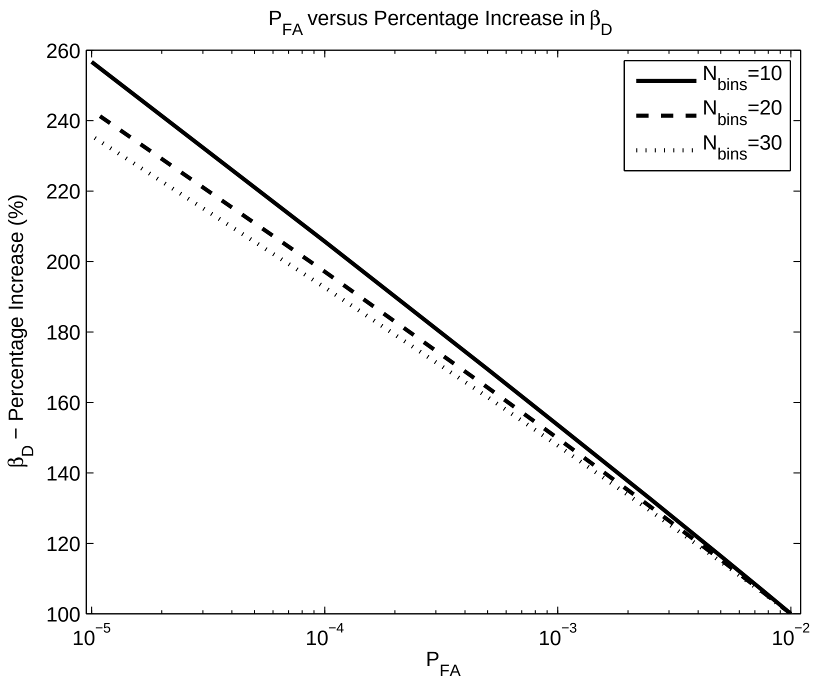

Figure 4.9: Change in β_D as P_{FA} decreases

Lowering the false positive rate by modifying the value of χ^2_D, as done above, can work well when the initial T_D is low. However, as we saw in the shrew attack scenario the values of T_D were quite large due to high subsampling rates, and longer periods being detected. When T_D is already large increasing it further to reduce the false positive rate is not a useful option. For example, from Table 4.8 one of the values of T_D was 1020 seconds, multiplying this value by 1.6 to lower P_{FA} from 0.01 to 0.001 (as done above) would increase T_D to 1632 seconds, an increase of 10.5 minutes.

An alternative to increasing T_D to reduce the probability of false alarm is to use post processing on the detection decisions to reduce the number of false positives and the subsequent user interaction. For example, when we are detecting periodic anomalies, harmonics of the anomalies should also be detected. Sub-densities that produce a false positive are unlikely to have a harmonic multiple that also produces a false positive. Therefore, by checking that a sub-density *and* its harmonic both detect the anomaly we can potentially reduce false positives without any user interaction.

Another possible method for post processing is to use some known characteristics of the anomaly, specifically how the anomaly manifests in the renewal density. As shown in Figures 4.5 and 4.8 both of the anomalies produce peaks in the renewal density. Therefore, a potential post processing technique would be to examine the values in sub-densities where an anomaly was detected. If detection was triggered by values in the renewal density estimate that were greater than the smooth approximation, then this is likely due to an anomaly. However, if the differences between the renewal density estimate and smooth approximation are both positive and negative this is more likely a false positive caused by Internet traffic alone.

Therefore, by using a combination of altering system settings (χ^2_D and T_D) and post processing on detection decisions it is possible to reduce the amount of false positives generated by our system, which makes our system require less user interaction to check the accuracy of our detection decisions.

4.6 Conclusion

In this chapter we examined three different applications of the IA^{2}D detection system. The first was detecting periodic events mixed with Internet traffic measurements. In this application we used renewal theory to present a thorough analysis of the detection system leading to an expression for the time-to-detection of a periodic event with period P. We then showed that the time-to-detection of our system was comparable to the state of the art bPDM system using theoretic analysis, and further showed that our system was capable of something bPDM was not, distinguishing between multiple period events. Finally, we applied IA^{2}D to real Internet traffic in a different scenario, and showed that, with some modification to the time-to-detection to account for the difference between a Poisson process and real Internet traffic, our system was able to reliably detect various periodic anomalies.

The second application we considered was detecting low-rate TCP-targeted DoS attacks, or shrew attacks. This type of attack was quite different from the periodic events we detected in the first application. In particular the period of a shrew attack is much longer than those we considered for the periodic event detection, however, since the shrew attack is designed to send bursts of packets at these periodic intervals we were still successful in detecting them in a reasonable amount of time. Another difference was that the longer inter-burst period presented a different challenge for estimating the renewal density used in detection. Namely, much higher downsampling rates were required to prevent false positives, this being a consequence of the Internet traffic not being an ideal Poisson process which was exaggerated at the longer inter-arrival times.

Chapter 5:

Multi-level IA^{2}D System

In this chapter we consider a different system architecture for our IA^{2}D detection system that we call the multi-level detection system. In this implementation, multi-level describes how detection is applied to the same Internet traffic measurements repeatedly, however, using different subsampling rates at each application. The purpose of the first level is to quickly determine if the Internet traffic measurements "likely" contain an anomaly. We use the term likely because at the first level the focus is on quickly processing the data, therefore, high subsampling rates are used and we use a lower threshold, χ^2_D, which generates more false positives results. If the measurements at the first level were determined to likely contain an anomaly then detection proceeds to the second level. At the second level we use a lower subsampling rate, and after reapplying IA^{2}D to the same measurements we are able to increase the confidence that indeed the measurements contain an anomaly and the result at the first level was not a false positive. By lowering the subsampling rate at the second level reprocessing the same measurements takes more processing time, however, the goal of the multi-level system is that by quickly performing analysis at the first level and only reprocessing at the second level infrequently the overall processing requirement is lowered.

Therefore, in this chapter we begin with a description of the multi-level detection system and discuss some potential applications in Section 5.1. Then starting in Section 5.2 we present a detailed case study of one particular application of the multi-level detection system, which we call the distributed forensic detection system. The distributed forensic detection system is motivated by the fact that much success has been achieved in detecting Distributed Denial-of-Service (DDoS) attacks at the target where multiple attack streams converge, however, methods attempting to trace the individual attack streams back toward their source have thus far been unsuccessful. In order to trace individual attack streams it is necessary to obtain measurements at multiple locations in the network, therefore, we propose a distributed measurement system. The distributed measurement system uses subsampling to reduce the storage cost to store measurements of the incoming packet stream. We combine the distributed measurement system with the multi-level version of our IA^{2}D detection system, which allows the data from the multiple measurement locations to be processed quickly at the first level, and only measurements from locations deemed to contain an attack will be reprocessed at the second level. Therefore, using our multi-level implementation we can increase the efficiency of having to process data from multiple recording nodes that is required to trace the individual DDoS attack streams back towards their source.

5.1 Multi-level Detection Approach

A diagram of the multi-level detection system is shown in Figure 5.1. Rather than discuss how the multi-level system operates on a single stream of input measurements, we present the system operation on the more general case, shown in Figure 5.1, where there are multiple recording nodes and that recording and processing occur in two separate locations. We analyze this implementation because it is the one we use in Section 5.2 to implement our distributed forensic detection system. Conveniently, analyzing the multi-level system in this configuration is actually quite general. For example, assume that instead of having multiple recording nodes our system only has a single recording node. With this assumption we can think of the data from each recording node in Figure 5.1 as representing a different recording period from our single recording node, e.g., recording node i records data from hour i in a 24 hour period. We do not assume that anomalies will be constant during an entire 24 hour period, so the anomalies will not appear in the data from all the recording nodes (during a single 24 hour period, that is, we do not assume that anomalies occur only a certain times of day). This scenario is similar to our forensic detection system where we assume that anomalies will not be present in all recording nodes at once. Therefore, all the analysis that we conduct in this section, while it is derived for multiple recording nodes, can be interpreted for a single recording node. Also, note that recording and processing can be done in the same machine. In this case no data transmission between recording and processing nodes is needed, and the analysis of the reduction in data transmission cost considered below would not apply.

We also assume throughout this section that the length of the data being processed at each level of the multi-level system is the same. As was shown in Section 4.3.1.1, normally when we increase the subsampling rate we also increase the time-to-detection, i.e., the length of data being processed. In the multi-level scenario, however, we increase subsampling to speed processing, and if we increased the length of the data being processed all the gains achieved by subsampling would be lost. Therefore, we fix the length of data being processed, which consequently lowers our confidence in detection at the first level of detection. This lower confidence in detection generates more false positives at the first level, however, we use the second level of detection to increase our confidence in detection and remove the false positives that passed the first level of detection.

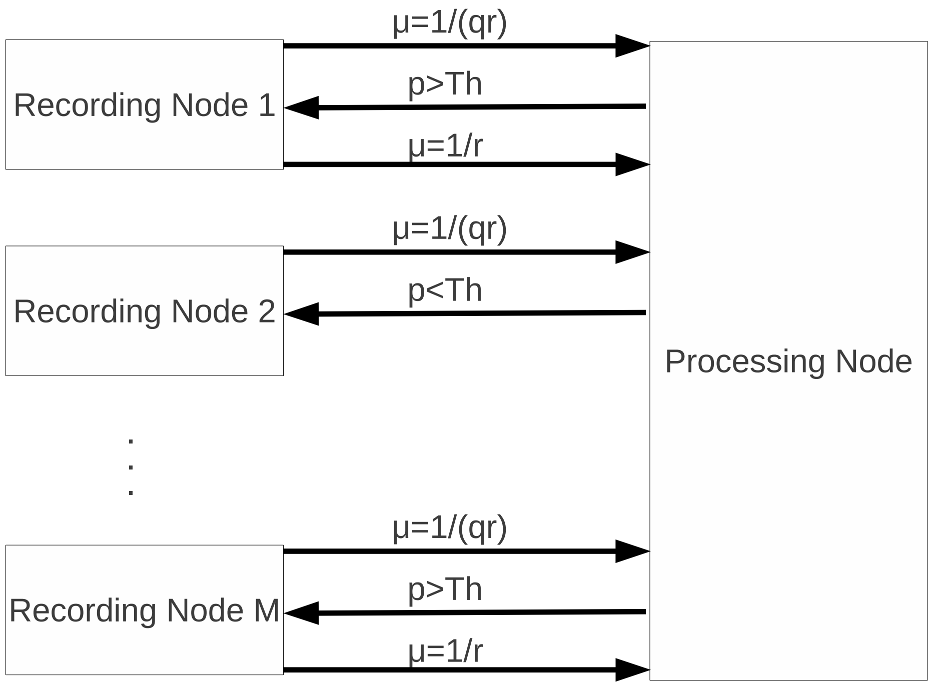

Figure 5.1: Transmission of data between recording and processing nodes in multi-levelscheme

Detection in the multi-level system proceeds as follows. We assume that the recording nodes record incoming packets measurements using some base subsampling rate, $\frac{1}{r}$ in Figure 5.1. Then for the first level additional subsampling is applied. Increasing the subsampling rate, given that the signal was originally subsampled using the method derived in Section 2.3, i.e., random subsampling, can be accomplished by applying the same subsampling procedure to the measurements already recorded. For example, if the incoming packets are initially subsampled by using the probability profile $l \sim Geo(r)$ to generate the measurements $X[k]$, which have a subsampling rate equal to $1/r$. Then a new set of measurements, $\hat{X}[m]$ is obtained by subsampling the measurements $X[k]$ using a new pmf, $\hat{l} \sim Geo(q)$, with corresponding subsampling rate $1/q$. The overall subsampling rate, from the input traffic to the measurements

$\hat{X}[m]$, is the product of the two subsampling rates, i.e., $\frac{1}{q \cdot r}$.

The measurements, $\hat{X}[m]$, are transmitted to the processing node (assuming recording and processing is done separately) and then processed by the detection system. After detection is performed the output is what we term the "confidence in detection", which is given by the Pearson statistic, p, described in Section 3.3.2. The Pearson statistic is related to the probability of false alarm by the relation, $p = 1 - P_{FA}$. In other words if one accepts the alternative hypothesis, that an anomaly is present, based on a particular value of p then the probability of a false positive is $1 - p$. Note that we actually have a vector of Pearson statistics, one for each sub-density. After processing, the vector of Pearson statistics is sent back from the processing node to the recording nodes. If, for the data in recording node i, the value of p in any sub-density exceeds some threshold Th, then that recording node retransmits the data back to the processing node at a lower subsampling rate $\mu = \frac{1}{r}$, and the data is reprocessed at the second level to increase the confidence that an anomaly is present.

With the multi-level implementation savings can be achieved in the two areas: (i) data transmission, and (ii) data processing. In the following sections we analyze the potential savings in these areas.

5.1.1 Data Transmission

The first savings is in the cost to transmit the data from the recording node to the processing node, assuming that processing is not done locally. Assume that each recording node records measurements with the base subsampling rate, $\frac{1}{r}$, and this results in blocks of data to be processed that are Υ gigabytes (GB) each. A block of data represents some fixed length of time T, so for each length of time T each measurement node transmits its block of data. If, as shown in Figure 5.1, we have M recording nodes then the cost to transmit the data, for one length of time T, is $M \cdot \Upsilon$ without additional subsampling, and $q \cdot M \cdot \Upsilon$ with additional subsampling, i.e., for level one of the multi-level implementation. The total cost to transmit the data from the recording nodes to the processing node for both levels of detection, during one length of time T, depends on the number of nodes that require additional processing, N_{tot}, where N_{tot} is the combination of nodes that contain an anomaly, N_{anom} and those that are due to false positives at the first level:

$$N_{tot} = N_{anom} + (M - N_{anom}) \cdot P_{FA} \tag{5.1}$$

and the total transmission cost, Υ_{tot}, for both stages is then given by:

$$\Upsilon_{tot} = q \cdot M \cdot \Upsilon + N_{tot} \cdot \Upsilon \tag{5.2}$$

To analyze the potential savings in data transmission we first consider the scenario when none of the recording nodes have anomalies, i.e., $N_{anom} = 0$. In this scenario the value of Υ_{tot} completely depends on the value of P_{FA} caused by processing data at the first level with a high subsampling rate. With $N_{anom} = 0$ the value of Υ_{tot} becomes:

$$\Upsilon_{tot} = q \cdot M \cdot \Upsilon + M \cdot P_{FA} \cdot \Upsilon = M \cdot \Upsilon\,(q + P_{FA}) \tag{5.3}$$

Therefore, in this scenario, we achieve a savings in transmission cost as long as $\Upsilon_{tot} < M \cdot \Upsilon$ which occurs when $P_{FA} + q < 1$ or $1 - q > P_{FA}$.

The converse scenario, where all nodes contain an anomaly *in the same block of time* T, i.e., $N_{anom} = M$, actually increases the data transmission cost because in this case $\Upsilon_{tot} = (q+1) \cdot M \cdot \Upsilon$. However we assume this scenario is unlikely, and instead we can determine the maximum value of N_{anom} for which we still achieve transmission savings. Rearranging the terms in (5.2) we get the expression:

$$\begin{aligned} \Upsilon_{tot} &= q \cdot M \cdot \Upsilon + (N_{anom} + (M - N_{anom}) \cdot P_{FA}) \cdot \Upsilon \\ \Upsilon_{tot} &= \Upsilon \cdot [M \cdot (q + P_{FA}) + N_{anom} \cdot (1 - P_{FA})] \end{aligned} \tag{5.4}$$

We achieve transmission gains as long as the term in brackets is less than M, which after some manipulation we can express as:

$$\begin{aligned} M &> M \cdot (q + P_{FA}) + N_{anom} \cdot (1 - (q + P_{FA}) + q) \\ N_{anom} &< M \left(\frac{1 - (q + P_{FA})}{1 - P_{FA}} \right) \end{aligned} \tag{5.5}$$

Thus, for a given detection scenario we know the values of q and P_{FA}, and can determine the maximum value of N_{anom}, i.e., the maximum number of recording nodes per length of time T that can contain an anomaly, where we still achieve savings in the cost to transmit the data.

5.1.2 Data Processing

The second area where we can make potential savings by using the multi-level implementation is in terms of data processing. Algorithmically the bottleneck in our detection algorithm is computing the inter-arrival times for the renewal density estimate, so this is where we will measure the data processing load. For each measurement, inter-arrival times between measurements must be computed until either the maximum inter-arrival time, $Maxx$, or the maximum inter-arrival order, n, has been reached. When we subsample by a factor μ this reduces the total number of measurements by the same factor μ. Further, the number of of inter-arrival times that must be computed before we reach either the maximum inter-arrival time or the maximum inter-arrival order is reduced again by a factor of μ. Overall, subsampling leads to an order of μ^2 reduction in the number of inter-arrival computations and thus roughly the same reduction in overall processing costs.

Following a similar derivation as was done for the data transmission cost, we assume that the number of processing operations required to process the measurements, for one block of time T, with the base subsampling rate, $\frac{1}{r}$, is equal to ξ operations (OPS). Therefore to process the data from the M recording nodes the processing cost is $M \cdot \xi$ OPS without additional subsampling, and $q^2 \cdot M \cdot \xi$ OPS with additional subsampling, i.e., for level one of the multi-level implementation. The total cost of processing the data from the recording nodes for both levels depends on the number of nodes that require additional processing, N_{tot}, which was given in (5.1). The total computation cost, ξ_{tot}, is then given by:

$$\xi_{tot} = q^2 \cdot M \cdot \xi + N_{tot} \cdot \xi \tag{5.6}$$

Notice that the total computational cost (5.6) is nearly identical to data transmission cost (5.2) except for the fact that the subsampling term is now squared, i.e., q^2.

As was done previously to analyze the potential savings in computational cost we first consider the scenario when none of the recording nodes have anomalies, i.e., $N_{anom} = 0$. Since the value of q^2 is the only difference in equations (5.6) and (5.2) we can omit the details of the derivation (see (5.3)) and conclude that we achieve a savings in computational cost as long as $P_{FA} + q^2 < 1$ or $1 - q^2 > P_{FA}$.

Just as in the data transmission case the converse scenario, where all nodes contain an anomaly, $N_{anom} = M$, increases the computational cost. So we will determine the maximum value of N_{anom} for which we still achieve computational savings. Following the derivation in (5.4) using q^2 and we arrive at the following result for the restriction on N_{anom} where we still accomplish computational savings.

$$N_{anom} < M\left(\frac{1-(q^2+P_{FA})}{1-P_{FA}}\right) \tag{5.7}$$

Just like when computing the transmission cost we can, for a given detection scenario with known values of q and P_{FA}, determine the maximum value of N_{anom} where we still achieve savings in data processing.

Unfortunately, because of there is no closed form expression that relates P_{FA} to q we are unable to determine the optimal operating point for our system. However, under normal operation conditions we would expect anomalies to occur infrequently, so it is more important to consider the transmission and processing cost when no anomalies are present. Since the values of q and P_{FA} are linked, i.e., selecting a smaller value of q (corresponding to a larger subsampling rate $1/q$) increases P_{FA}, if given multiple choices for the value of q and P_{FA} in a detection scenario we select the values of q and P_{FA} that minimize the transmission cost $P_{FA} + q < 1$ and the processing cost $P_{FA} + q^2 < 1$. Then when an attack occurs we can use equations (5.5) and (5.7) to determine the value of N_{anom} when we have to begin modifying our implementation, i.e., when to modify the subsampling rate $1/q$ or when to stop the multi-level implementation in favor of the standard IA^{2}D implementation.

5.2 Distributed Forensic Detection System

For the rest of the chapter we consider the use of our multi-level version of IA^{2}D in a particular detection scenario that we call distributed forensic detection. Anomaly and, in particular, Denial-of-Service (DoS) attack detection at or near the target has been studied extensively [6, 27, 52, 74]. Traffic or flow statistics, e.g., traffic volume or flow entropies [27, 74], near the target are altered sufficiently to signal a change in traffic behavior indicating the presence of DoS traffic. Many detection schemes are currently available that can quickly detect anomalies at the target and allow rapid action to be taken limiting the damage done by the attack. However, once an attack has been discovered most detection systems return to normal functioning waiting for the next attack to occur. Forensic analysis, looking to answer when and where the attacks originated, is typically left up to network managers to search through network logs or measurements of network traffic near the target. Such analysis is time consuming, and even if it is possible to determine which measured packets contributed to the attack, information about its source, e.g, IP address, is generally unreliable [68]. Thus tracing the attack traffic back towards its source is difficult.

Tracing the path of attack traffic is even more challenging for Distributed DoS (DDoS), and low-rate TCP-targeted DoS attacks [46]. Tracing these attacks is difficult because, while the combination of multiple streams might be noticeable near the target, at other points in the network the individual streams are at a lower rate and are less obvious. Detecting the individual streams generally requires more data to be collected over a longer duration before enough attack packets are observed to be detectable. Additionally, while the combined attack might be detected quickly, some of the individual streams might be active for much longer durations going unnoticed. Therefore tracing multiple attack streams requires a system where multiple nodes record packet information over long durations, not just when an attack is detected, in order to increase the likelihood of detecting the low-rate individual streams. As an example consider Figure 5.2, where the DDoS attack might be easily discoverable at the target due to the convergence of multiple streams of attack traffic (labeled A_i), all much smaller than the combined stream. Following the detection of the DDoS attack it would be desirable to trace the path of the individual attack streams (the A_i) back through the network as close as possible to their origin (the zombies). However, the high cost of storing measured packet information makes storing long segments of traffic data prohibitively expensive.

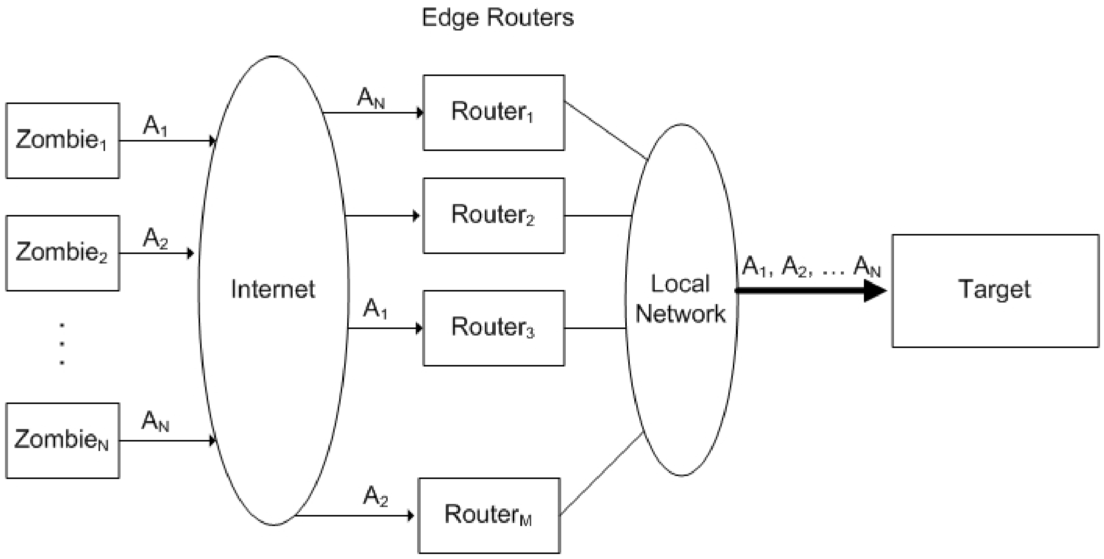

Figure 5.2: DDoS attack converging at target

Motivated by this, we propose a distributed measurement system designed for forensic anomaly detection. The main contribution is a system design which minimizes the storage size of trace data by using our *SigVec* signal representation (Section 2.2.4), and employing our optimized subsampling technique (Section 2.3) to only record measurements for a subset of incoming packets. The signal representation reduces the storage rate by at least 80% per packet over standard ERF or PCAP file types, while subsampling further reduces storage cost by a factor equal to the subsampling rate. As an example, for the dataset we used in our simulations, which is the same LANDER dataset used in Section 4.3.2, the average data rates were around 300 Mbps [20]. Storing PCAP measurements for 60 seconds worth of traffic requires approximately 1 GB of storage (more for ERF). Using standard GZIP compression reduces the storage requirement for the PCAP data by a factor of around 5, yet this still requires 12 GB worth of data per hour. Our signal representation, *SigVec* derived in Section 2.2.4, reduces storage by a factor 5 in both compressed and uncompressed formats over PCAP (a factor of 6 over ERF), with savings of an additional factor of 10 or more possible via subsampling. Moreover, we show that these storage savings can be achieved with essentially no penalty on detection accuracy.

To enhance the efficiency of our distributed measurement system, we also propose to use the multi-level version of IA^{2}D to expedite anomaly detection Thus the main contribution of this work is to combine the multi-level detection mechanism that we described in Section 5.1 with a distributed measurement system that minimizes storage cost as discussed in Section 5.3. Experiments in Section 5.4 show that the multi-level detection method can be used at high subsampling rates to expedite processing of long segments of measured data while still detecting low rate DoS attacks.

5.2.1 Related Work

Compared to most previous work on tracing network attacks back to their source the key novelty in our work is to focus explicitly on reducing the amount of resources needed for these tasks in a distributed setting. Specifically we consider reductions in the cost of storage (which reduces the cost of each measurement node and allows more to be deployed), processing and transmission (since our multi-level processing allows only important data to be fully transmitted or processed). Some authors [5] proposed systems where routers store information about packets they forward, e.g., source MAC addresses, so that, after an attack is detected, the path of the attack can be traced by comparing the stored information with that in neighboring routers to find the path of the attack packet. Our system is different and, we claim, more general because we do not require being able to distinguish between attack packets and regular background traffic, which allows us to operate in situations, e.g., with encrypted traffic, where it is not possible to differentiate among the packets. Further, in our method processing is independent at each recording node, while other systems such as [5] require information to be stored at all routers, which for security and privacy reasons might not be possible.

There are a few existing methods which use time domain detection schemes similar to ours, but unlike in our work, limiting overall storage requirements is not a major concern. In [56], the authors find the strongest correlation of the data between the current router and the possible routers which could have forwarded the attack. Similar to our work this method must contend with the size of the attack traffic being very small compared to the background traffic at a router, which means that the correlation the attack traffic creates between routers will be small as well. Therefore, like our method, this technique likely requires very long segments of data in order to produce reliable results. One disadvantage of the system in [56] is that it requires measurements to be recorded at all routers, in order to find the largest correlation, whereas ours can operate independently at any router which records measurements.

Another similar system is the distributed low-rate TCP attack detection system discussed in [73]. Like our method, the detection scheme in [73] uses time domain signal processing methods (autocorrelation) to detect periodic behavior linked to certain anomalous TCP flows. Also, their method samples the incoming packet stream for efficiency, however, they generate a uniformly sampled signal representation, which as discussed in Section 2.2.4 is inferior to our signal representation for anomaly detection.

Both of these time domain detection systems use signal representations that are different from the one presented here. Potentially these systems could be modified to use measurements from our system allowing them to benefit from the reduced storage costs. Thus, while we present one specific detection method, the distributed measurement system is general and can potentially be used with alternate forensic detection methods.

5.3 Distributed Measurement and Processing System Architecture

Our measurement and processing system architecture is designed to be flexible in how it is implemented. One of the fundamental building blocks of our system are recording nodes, which are located at routers and record measurements of packets that are forwarded by that router. The other component of our system are one or more processing nodes, which receive data from the recording node and implement our detection strategy. Since our system is designed for forensics it is possible that a node could be recording incoming packets until an attack is detected elsewhere in the network, at which time the node could switch into a processing mode and perform detection on its own previously recorded data. It is also important to note that this system is designed to be compatible with the recommendations of the IETF PSAMP working group [3] for creating a flexible measurement system architecture. For instance, our recording node is designed to follow the recommendation for a PSAMP device. It includes an appropriate selection process (our sampling technique), and could follow the PSAMP recommendations on how to transfer packet information (exporting process) to a collector (our processing node). Therefore, implementation of our distributed measurement system should be possible in devices meeting the PSAMP recommendations. In addition to the up and coming PSAMP architecture, our system can also be implemented using hardware measurement systems such as those from ENDACE [24], or using lower cost alternatives like the software based measurement systems described in Section 2.2.

Given the building blocks of our system, the architecture with which the system is implemented is flexible to the constraints of the network under observation. Similar to the work in [73] the ideal implementation would be to have measurements recorded at each hop between the attack origin and the target. However, this may not be possible in general; certain operators may not allow measurements in all or part of their networks, while the large amounts of storage needed for measurements can limit the practicality of these approaches. Since we cannot guarantee access to data at all nodes, our detection mechanism is designed to work independently of data from other recording nodes, and thus can be used at any accessible observation points along the path from source to target. While this might not allow us to always trace the attack all the way back to the source it can, for instance, be used to trace the attack as far back as to an edge router of a local network as in [70].

5.4 Distributed Measurement and Multi-level Detection Simulation

To show the advantages of our combined distributed measurement system and multi-level detection we present the following detection scenario as a case study. Assume, as in Figure 5.2, that a DDoS attack has been detected at a link in the local network. Our goal is to trace the particular components of the attack (the A_i) at the edge routers of the local network. In this experiment the traffic at the edge routers will be simulated using a 25 minute long trace obtained by the LANDER Los Nettos trace collection system [20]. In practice each router experiences different traffic, however, we use one trace here to make the example simple. The average bit rate for the segment is 300 Mbps.

For our experiment assume that there are 10 edge routers, each receiving the 300 Mbps average bit rate. Further, we assume that some of our edge routers encounter very low-rate attacks with bit rates of 15 and 10 Mbps respectively. This corresponds to the attacks being less than 5% and 3.3% of the total combined bit rate, respectively. Note, these attack rates are small compared to those considered in the literature (e.g., [74]). These bit rates could correspond to the traffic generated by the zombie encountering a bottleneck, a 10 Mbps Ethernet link for instance, or could be due to limits in the maximum output rate from the zombie. If the attacks are generated using 1518 byte packets then the associated packet rate of the attacks are 1.6% and 1.1% of the total packet rate, respectively. The packet delay variation that occurs as the attack traverses the network from origin to target is replicated by adding jitter to the ideally periodic timing of the attack packets. The amount of jitter used in our simulations is approximately 25 microseconds.

5.4.1 Data Reduction via Subsampling and SigVec

Since our proposed system is designed to be deployed in a distributed manner the data storage requirement for collecting packet measurements increases with the number of recording nodes. Detecting low-rate anomalies requires very long segments of measurements to be recorded to gather enough evidence to reliably detect the anomaly, further increasing the required data storage cost.

To reduce the cost of data storage compared to standard PCAP and ERF data formats we propose the use of our signal representation, SigVec, discussed in Section 2.2.4 combined with the subsampling technique we derived in Section 2.3.

To show the benefit of our data reduction methods consider the storage and transmission requirements associated with the experiment described above. In Table 5.1 we compare the requirement to store, per recording node, a 25 minute trace segment in various formats. Compared to uncompressed PCAP format just using SigVec results in a reduction by a factor of 5, and compared to ERF using SigVec reduces data storage size by a factor of 6. Recording data at all 10 nodes using SigVec would reduce storage by 200 GB before downsampling. If we further downsample the measurements by a factor of 10, this reduces storage cost by a factor of 50 over PCAP, with minimum impact of detection.

Storage Format	Size - GB	Size - GB
	Uncompressed	Compressed
ERF	29.97	7.90
PCAP	24.88	5.69
SigVec No SS	4.51	0.93
SigVec $\mu = 10$	0.46	0.14
SigVec $\mu = 25$	0.18	0.06

Table 5.1: Storage requirements for detection example measurements

5.4.2 Multi-Level Detection Results

Figure 5.3 shows confidence in detection, or Pearson statistic p, for the three different attack scenarios plotted for various subsampling rates. In [52] it was shown that the background traffic generally scores below 50% for confidence in detection, so a value of 65% is a reliable threshold for p at the first level yet not good enough to guarantee detection. From Figure 5.3 we see that given the subsampling rate of 25 both the 5% and 3.3% attacks are detected with a confidence greater than 65%. Further, we can detect the attacks with a confidence greater than 99% with a subsampling rate of 10, therefore, we will use $p = .99$ as the threshold at our second level of detection. Therefore we choose to implement our multi-level detection system using the following parameters, as shown in Figure 5.1:

- Base subsampling rate - $1/r = 10$
- Additional subsampling rate - $1/q = 2.5$
- Threshold - $Th = 65\%$

With these parameters we have that the probability of false alarm at the first level of detection is $P_{FA} = 1 - p = 1 - Th = .35$ and $q = .4$.

Using the expressions we derived in Section 5.1 we can analyze the potential savings of our distributed detection system when no anomalies are present, and determine the maximum number of nodes that can contain an anomaly for us to still achieve savings. First, when no anomalies are present, we know from (5.3) that the data transmission savings achieved using the multi-level implementation compared to sending the data at the lower subsampling rate is given by $1 - (q + P_{FA})$, which for our example is .25 or 25%. Using the data from Table 5.1 for 10 recording nodes the cost to transmit the data at a subsampling rate of 10 is $10 \cdot 0.14 = 1.4$ GB. With the multi-level implementation we transmit $.25 \cdot 1.4 = .35$ GB less data. Next using (5.6) we determine that the that the computational cost savings is given by $1-(q^2+P_{FA})$, which in this example is $1 - (.4^2 + .35) = .49$ or 49%, a significant reduction.

Next, if we assume that anomalies are present in our system we can determine the maximum number of nodes, N_{anom}, that can contain an anomaly for us to still achieve savings in transmission and processing cost. Using (5.5) we compute the value of N_{anom} to be:

$$N_{anom} < M\left(\frac{1-(q+P_{FA})}{1-P_{FA}}\right) < 10\left(\frac{1-.75}{1-.35}\right) < 10\cdot .38 < 3.8 \tag{5.8}$$

and using (5.7) the value of N_{anom} is given by:

$$N_{anom} < M\left(\frac{1-(q^2+P_{FA})}{1-P_{FA}}\right) < 10\left(\frac{1-.51}{1-.35}\right) < 10\cdot .75 < 7.5 \tag{5.9}$$

Thus we see that we can still achieve savings in data transmission when up to 3 nodes have anomalies in them, and we save on computational cost even when 7 nodes have anomalies present.

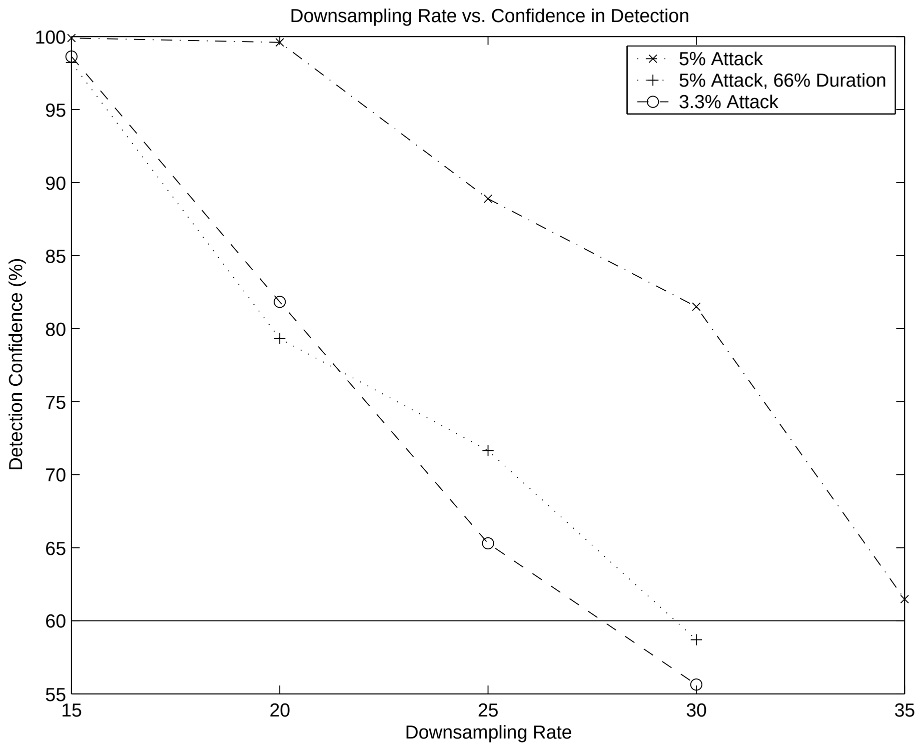

Figure 5.3: Downsampling rate vs. confidence in detection

In the previous experiment the attack traffic was active for the entire length of the recording, which in practice is highly unlikely especially given the long duration traces that the measurement system records. If the attack is only present during a portion of the analyzed trace the effect is similar to decreasing the attack rate by a proportion equal to the attack duration. For example, if the 5% attack is only active during 66% of the recorded trace, then our ability to detect the attack by analyzing the entire trace is similar to our ability to detect a $.66 \cdot 5\% = 3.3\%$ attack. This example is shown in Figure 5.3, and we see that our ability to detect the 5% attack with 66% duration is similar to the 3.3% attack. Therefore even if the attack is not present during the entire trace we retain the ability to detect the attack.

5.5 Conclusion

In this chapter we modified our IA^{2}D detection system and derived a different implementation that we called the multi-level detection strategy. The multi-level detection method processes recorded measurements at two levels. At the first level the data is highly subsampled, this allows IA^{2}Dto process the measurements efficiently and the goal of the first level is to determine measurements that 'likely' contain an attack. On traces determined to likely contain an attack more reliable detection is performed using a second level of the multi-level system with a lower subsampling rate. In Section 5.1 we determined expressions for the potential data savings and determined the system operating point when it was better to use the standard IA^{2}D implementation instead of the multi-level version.

As an example application we consider a distributed forensic analysis system that included multiple measurement nodes and a single processing node that implemented the multi-level version of IA^{2}D. We showed our measurement system could achieve an

80% reduction in storage cost compared to the more commonly used PCAP and ERF file formats. Finally, we showed that using our multi-level implementation we could detect very low-rate periodic anomalies with data transmission savings up to 25% and a 49% reduction in processing cost compared to the standard implementation of IA^{2}D.

Chapter 6:

Conclusions and Future Work

A method to integrate Internet measurement and analysis has been proposed by constructing a flexible signal representation, *SigVec*. The benefits of the *SigVec* signal representation are: (i) it is general, i.e., can be implemented in any existing measurement systems, (ii) allows for subsampling, which can be useful to reduce data storage, transmission and processing costs. This signal representation is the most natural for typical measurement systems, however, it is typically not used for analysis as most analysis techniques convert the event based measurements to a more common time series representation, in order to be compatible with standard signal processing techniques (e.g., fast Fourier transform, power spectral density, etc.). However, we showed that analysis techniques based on the *SigVec* signal representation could be designed to perform as well as existing techniques with some additional benefits, such as detecting anomalies in aggregate traffic without separating the traffic into flows.

In Chapter 2 we analyzed and derived a set of timing error models to represent a software based measurement system. The *SigVec* signal representation was selected because it was the natural method to mitigate the effect of these timing errors when combined with subsampling. Methods to optimize subsampling in the measurement system where discussed in Section 2.3. By optimizing subsampling it is possible to preserve the desired characteristics of the Internet traffic that are important for a specific type of analysis.

Using the signal representation and a formulation from renewal theory, called the renewal density, we derived a low-rate periodic detection system, IA^{2}D, in Chapter 3. The design of this system showed that it was possible to use the natural signal representation for Internet traffic analysis without having to convert to a time series representation. Additional benefits of our system design were its ability to distinguish between multiple periodic anomalies, something that is not possible in many existing systems.

Then in Chapter 4 we consider a number of applications of IA^{2}D. In each of these applications we used renewal theory to parameterize the system, and showed that, with modifications based on measurable Internet traffic characteristics, we could reliably select parameters for the detection system that produced reliable detection results for real Internet traffic.

Finally, in Chapter 5 we modified our IA^{2}D detection system to derive our multi-level detection system. The multi-level detection method processes recorded measurements at two levels. At the first level the data is highly subsampled, allowing measurements to be processed efficiently to determine collections of measurements that 'likely' contain an attack. Traces that were determined to likely contain an attack are then processed at a second level, with a lower subsampling rate, to increase detection reliability.

Thus, by selecting the natural signal representation, *SigVec*, and designing analysis methods that make use of this representation we showed that it is possible to integrate systems for the measurement and analysis of Internet traffic.

6.1 Future Work

Given our general *SigVec* signal representation and the design of IA^{2}D based on renewal theory much of the work contained herein can be applied to a wide range of signals interesting to scientific researchers, namely those that occur naturally as point processes. Examples of some naturally occurring point processes include the firing of neurons in the human brain, various forms of financial data (i.e., purchase and sale times of in the stock market), and computer network events not considered here, such as network alarms or flow arrivals [41, 60]. Typical behavior for many of these signals is often modeled by a renewal process, and deviations from typical behavior, i.e., anomalies, can signify health concerns in biological signals or potential malicious activity in stock market and computer network data, making their detection crucial.

Two signals of interest that are particularly relevant to our work are the timing of tweets on the popular social networking site 'Twitter', and network flow start times. The second case has been considered in detail, and in a similar context, in [7], where the goal of that work was to discover very low-rate periodic flows in an aggregate collection of flows. Periodic flows could be generated by normal computer user behavior, such as RSS feed aggregators polling sites periodically searching for new content, however, the periodic flows could also represent malicious behavior like zombies computers in a Botnet periodically checking for instructions from the zombie master waiting for an attack to be unleashed. Given that our detection system is designed to detect long-term periodic activity this application is a good fit for IA^{2}D.

On this application we have performed some preliminary analysis. The analysis was done by taking existing Internet traffic measurements, and filtering the measurements such that we were left with only packets with both the "SYN" and "ACK" flags set to 1. The SYN and ACK flags should only be set to 1 in a packet header when the server is responding to a client who has initiated a TCP session; this is the second step of the three-way handshake used by the TCP protocol [7]. Packets with both the SYN and ACK flags set to 1 are a good indication of the start of a TCP session, or a flow.

We examined the renewal density estimate obtained from the filtered set of Internet traffic measurements, and observed periodic activity. However, certain problems exist with the periodic activity observed that require more research before this scenario can be studied further. One problem is that the level of periodic activity observed does not correspond to the observation window. In other words the number of periodic inter-arrivals observed do not correspond to how many intervals of that period occur during the observed time period. The likely cause of this is either the periodic activity is quasi-periodic (random multiple of the base period, see Section 4.3.3), or the source of the periodic activity was only active during a portion of the observation window. Another problem is that we have yet to determine the cause of the observed periodic activity. Therefore, while we have observed periodic inter-arrival times in measurements of flow start times more research needs to be done to determine the cause of the observed periods as well as to determine a suitable method to determine the time-to-detection for the periodic activity observed.

Recently, Twitter has become increasingly popular and increasingly lucrative as companies and advertisers find new ways to utilize the micro-blogging site. However, with the increase in legitimate users and revenue being generated comes the increase in malevolent use on the site as well. A large percentage of user accounts are created each day by spammers, who flood the site with links to advertisements or even worse, like sites that attempt to install computer viruses on unsuspecting users. The site managers spend considerable time and resources attempting to sort out spammer accounts from legitimate users, and even when a spammer account is shut down another quickly takes its place. If the behavior of spammer activity on the site can be distinguished from regular user activity then the detection of spam could be automated saving the site managers time and money. We have recently looked into methods to identify spammer behavior on Twitter, and preliminary analysis indicates that some high volume accounts tend to generate tweets at periodic intervals. For example, we generated a Twitter account and followed approximately 150 normal users, and 1 high volume account that was selected because it was used primarily to sell real estate. After recording the timestamps for all the tweets received on the generated Twitter account during a 200 hour interval, we estimated the renewal density from the data, which is shown in Figure 6.1. Clearly, there is a strong periodic component in the renewal density at intervals of 300 seconds (5 minutes). It was verified that the periodic component was largely due to the high volume, real estate salesperson's Twitter account. Therefore, Twitter could represent another potential application of our IA^{2}D detection system.

One concern with using IA^{2}D to detect periodic messages on Twitter is that detecting periodic tweets alone might not be enough to determine spammer activity. For example, certain automated yet legitimate users, i.e., news websites, could generate tweets at periodic intervals and not be considered spam at all. Therefore, in order to reliably sort spam from the mountains of legitimate user data will require additional metrics beyond what IA^{2}D can offer, however, its use in combination with post-processing methods could from a powerful automated detection system.

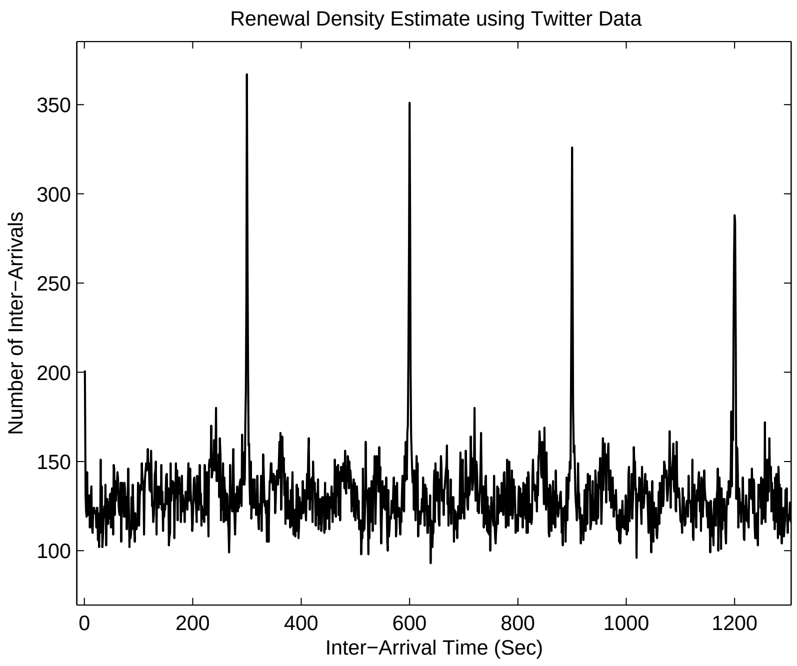

Figure 6.1: Renewal density estimate from Twitter data

References

[1] Abilene-III trace dataset. Provided Online by WAND Research Group, University of Waikato - *http://www.wand.net.nz/wits/ipls/3/*.

[2] IETF IPFIX Working Group. charter available online at *http://datatracker.ietf.org/wg/ipfix/charter*.

[3] IETF PSAMP Working Group. charter available online at *http://datatracker.ietf.org/wg/psamp/charter*.

[4] P.D. Amer and L.N. Cassel. Management of sampled real-time network measurements. In *Local Computer Networks, 1989., Proceedings 14th Conference on*, pages 62–68, Oct 1989.

[5] T. Baba and S. Matsuda. Tracing network attacks to their sources. *IEEE Internet Computing*, 6(2):20–26, 2002.

[6] P. Barford, J. Kline, D. Plonka, and A. Ron. A signal analysis of network traffic anomalies. In *IMW '02: Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment*, pages 71–82, New York, NY, USA, 2002. ACM.

[7] G. Bartlett, J. Heidemann, and C. Papadopoulos. Using low-rate flow periodicities for anomaly detection: Extended. Technical Report ISI-TR-2009-661, USC/Information Sciences Institute, August 2009.

[8] L.S. Brakmo and L.L. Peterson. TCP vegas: End to end congestion avoidance on a global internet. *IEEE Journal on Selected Areas in Communications*, 1995.

[9] R. L. Carter and M. E. Crovella. Measuring bottleneck link speed in packet-switched networks. Technical report, Performance Evaluation, 1996.

[10] Y. Chen and K. Hwang. Tcp flow analysis for defense against shrew ddos attacks. In *IEEE International Conference on Communications 2007 ICC 07*, pages 1–8, 2007.

[11] Y. Chen, K. Hwang, and Y.-K. Kwok. Filtering of shrew ddos attacks in frequency domain. In *The IEEE Conference on Local Computer Networks*, pages 8 pp. –793, Nov 2005.

[12] C.M. Cheng, H.T. Kung, and K.S. Tan. Use of spectral analysis in defense against dos attacks. In *Global Telecommunications Conference, 2002. GLOBECOM '02. IEEE*, volume 3, pages 2143–2148 vol.3, Nov. 2002.

[13] J. Cheng, D.X. Wei, and S. H. Low. FAST TCP: Motivation, architecture, algorithms, performance. In *Proceedings of IEEE INFOCOM*, pages 2490–2501, 2004.

[14] S. Y. Chun and A. Shapiro. Normal versus noncentral chi-square asymptotics of misspecified models. *Multivariate Behavioral Research*, 44:803–827, 2009.

[15] Cisco Systems, Inc. Introduction to Cisco IOS NetFlow. Technical report, Availabile online at *http://www.cisco.com/*, 2007.

[16] K. Claffy, G. Polyzos, and H.W. Braun. Application of sampling methodologies to network traffic characterization. *SIGCOMM Computer Communication Review*, 23(4):194–203, 1993.

[17] D. Cousins, C. Partridge, K. Bongiovanni, A.W. Jackson, R. Krishnan, T. Saxena, and W.T. Strayer. Understanding encrypted networks through signal and systems analysis of traffic timing. In *Aerospace Conference, 2003. Proceedings. 2003 IEEE*, volume 6, pages 2997–3003, 8-15 2003.

[18] D. Cousins, C. Partridge, A. W. Jackson, R. Krishnan, T. Saxena, and W. T. Strayer. Using signal processing to analyze wireless data traffic. In *Proc. ACM workshop on Wireless Security*, pages 67–76, 2002.

[19] M. Dashtbozorgi and M.A. Azgomi. A high-performance software solution for packet capture and transmission. *Computer Science and Information Technology, International Conference on*, 0:407–411, 2009.

[20] Scrambled Internet Trace Measurement dataset. PREDICT ID USC-LANDER/lander_sample-20080903. Provided by the USC/LANDER project. *http://www.isi.edu/ant/lander*, Traces taken for the full day on 2008-09-03.

[21] L. Råde. Theorems for thinning of renewal point processes. *Journal of Applied Probability*, 1972.

[22] C. Demichelis and P. Chimento. IP packet delay variation metric for ip performance metrics IPPM. Technical Report RFC 3393, The Internet Society, 2002.

[23] S. Donnelly. *High Precision Timing in Passive Measurements of Data Networks.* PhD thesis, The University of Waikato, 2002.

[24] S. Donnelly. Endace DAG time-stamping whitepaper. Technical report, Endace Technology Ltd., 2007.

[25] A.B. Downey. Using pathchar to estimate internet link characteristics. In *SIGMETRICS '99: Proceedings of the 1999 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pages 222–223, New York, NY, USA, 1999. ACM.

[26] N. Duffield, C. Lund, and M. Thorup. Estimating flow distributions from sampled flow statistics. *IEEE/ACM Trans. Netw.*, 13(5):933–946, 2005.

[27] L. Feinstein, D. Schnackenberg, R. Balupari, and D. Kindred. Statistical approaches to DDoS attack detection and response. In *DARPA Information Survivability Conference and Exposition, 2003. Proceedings*, volume 1, pages 303–314 vol.1, April 2003.

[28] D. Freedman and P. Diaconis. On the histogram as a density estimator: l_2 theory. *Probability Theory and Related Fields*, 57:453–476, 1981. 10.1007/BF01025868.

[29] E. Frees. Nonparametric renewal function estimation. *The Annals of Statistics*, 14, 1986.

[30] Alberto Leon Garcia. *Probability and Random Processes for Electrical Engineering.* Addison Wesley, 2nd edition, 1994.

[31] P.E. Gill, W. Murray, and M.H. Wright. *Practical Optimization.* Academic Press, 1981.

[32] A. Grad and H. Solomon. Distribution of quadratic forms and some applications. *The Annals of Mathematical Statistics*, 26:464–477, 1955.

[33] P. Greenwood and M. Nikulin. *A Guide to Chi-Square Testing.* Wiley Series in Probability and Statistics, 1996.

[34] X. He. *Detecting Periodic Patterns in Internet Traffic with Spectral and Statistical Methods.* PhD thesis, University of Southern California, 2006.

[35] A. Hussain, J. Heidemann, and C. Papadopoulos. Identification of repeated DoS attacks using network traffic forensics. Technical Report ISI-TR-2003-577b, USC/Information Sciences Institute, August 2003. Originally released August 2003, updated June 2004.

[36] A. Hussain, J. Heidemann, and C. Papadopoulos. Identification of repeated denial of service attacks. In *Proceedings of the IEEE Infocom*, Barcelona, Spain, April 2006.

[37] Intel. Interrupt moderation using Intel gigabit ethernet controllers. Technical report, Intel Corporation, April 2007.

[38] Sunita Jain. Spartan-6 FPGA connectivity targeted reference design performance. Technical report, Xilinx, 2010.

[39] P. Kamath, K.-C. Lan, J. Heidemann, J. Bannister, and J. Touch. Generation of high bandwidth network traffic traces. In *Proceedings of the 10th International Workshop on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems*, Fort Worth, TX, 2002.

[40] T. Karagiannis, M. Molle, M. Faloutsos, and A. Broido. A nonstationary Poisson view of internet traffic. In *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, volume 3, pages 1558 –1569 vol.3, 7-11 2004.

[41] D. Katabi, I. Bazzi, and Y. Xiaowei. A passive approach for detecting shared bottlenecks. *Proceedings of the Tenth International Conference on Computer Communications and Networks, 2001.*, pages 174–181, 2001.

[42] D. Katabi and C. Blake. Inferring congestion sharing and path characteristics from packet interarrival times. Technical report, LCS, 2001.

[43] K. Keys, D. Moore, R. Koga, E. Lagache, M. Tesch, and K. Claffy. The architecture of coralreef: An internet traffic monitoring software suite. Technical report, CAIDA: Cooperative Association for Internet Data Analysis, University of California San Diego, 2001.

[44] I. Kim, J. Moon, and H. Y. Yeom. Timer-based interrupt mitigation for high performance packet processing. In *Proc. 5th International Conference on High-Performance Computing in the Asia-Pacific Region, Gold*, 2001.

[45] S. Kullback and R. A. Leibler. On information and sufficiency. *The Annals of Mathematical Statistics*, 1951.

[46] A. Kuzmanovic and E. W. Knightly. Low-rate TCP-targeted denial of service attacks: the shrew vs. the mice and elephants. In *Proceedings of SIGCOMM*, pages 75–86, New York, NY, USA, 2003. ACM.

[47] K. Lai and M. Baker. Measuring bandwidth. In *INFOCOM '99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 1, pages 235 –245 vol.1, mar 1999.

[48] K. Lai and M. Baker. Nettimer: A tool for measuring bottleneck link bandwidth. In *Proceedings of the USENIX Symposium on Internet Technologies and Systems*, pages 123–134, 2001.

[49] S. Ma and J.L. Hellerstein. Mining partially periodic event patterns with unknown periods. pages 205 –214, 2001.

[50] C. L. T. Man, G. Hasegawa, and M. Murata. ICIM: An inline network measurement mechanism for highspeed networks. In *4th IEEE/IFIP Workshop on End-to-End Monitoring Techniques and Services*, pages 66–73, 2006.

[51] N. Markovich and U. Krieger. Nonparametric estimation of the renewal function by empirical data. *Stochastic Models*, 22, 2006.

[52] S. McPherson and A. Ortega. Analysis of internet measurement systems for optimized anomaly detection system design. Technical Report 0907.5233, Arxiv, July 2009.

[53] S. McPherson and A. Ortega. Improved internet traffic analysis via optimized sampling. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2010.

[54] U. Mitra, A. Ortega, J. Heidemann, and C. Papadopoulos. Detecting and identifying malware: A new signal processing goal. *IEEE Signal Processing Magazine*, 23(5):107–111, September 2006.

[55] Randolph Nelson. *Probability, Stochastic Processes and Queuing Theory: The Mathematics of Computer Performance Modeling.* Springer-Verlag, 1995.

[56] K. Ohta, G. Mansfield, Y. Takei, N. Kato, and Y. Nemoto. Detection, defense, and tracking of internet-wide illegal access in a distributed manner. In *Proceedings of INET 2000*, 2000.

[57] V. Paxson and M. Allman. Computing TCP's retransmission timer. Technical Report RFC 2988, The Internet Society, 2000.

[58] V. Paxson and S. Floyd. Wide-area traffic: The failure of poisson modeling. *IEEE Transactions on Networking*, 61(4):215–225, 1995.

[59] S. Pemmaraju and S. Skiena. *Computational Discrete Mathematics.* Cambridge University Press, 2003.

[60] Donald H. Perkel, George L. Gerstein, and George P. Moore. Neuronal spike trains and stochastic point processes. II. simultaneous spike trains. *Biophys. J*, 7:419–440, 1967.

[61] R. Prasad, M. Jain, and C. Dovrolis. Effects of interrupt coalescence on network measurements. In *Proceeding of Passive Active Measurement (PAM) Workshop*, pages 247–256, 2004.

[62] A. Rényi. A characteristization of poisson processes. *Magyar Tud. Akad. Mat. Kutaló Int. Közl*, 1956.

[63] K. Salah. On the accuracy of two analytical models for evaluating the performance of gigabit ethernet hosts. *Information Sciences*, 176:3735–3756, 2006.

[64] K. Salah. To coalesce or not to coalesce. *International Journal of Electronics and Communications*, 61(4):215–225, 2007.

[65] K. Salah and K. El-Badawi. On modeling and analysis of recieve livelock and CPU utilization in high-speed networks. *Internation Journal of Computers and Applications*, 2006.

[66] K. Salah, K. El-Badawi, and F. Haidari. Performance analysis and comparison of interrupt-handling schemes in gigabit networks. *Comput. Commun.*, 30(17):3425–3441, 2007.

[67] M. Sankaran. On the non-central chi-square distribution. *Biometrika*, 46:235–237, 1959.

[68] S. Savage, D. Wetherall, A. Karlin, and T. Anderson. Practical network support for ip traceback. *SIGCOMM Comput. Commun. Rev.*, 30(4):295–306, 2000.

[69] J. Shafer and S. Rixner. A reconfigurable and programmable gigabit ethernet network interface card. Technical Report TREE0611, Rice University - Department of Electrical and Computer Engineering, 2006.

[70] A. Shevtekar, K. Anantharam, and N Ansari. Low rate TCP denial-of-service attack detection at edge routers. *IEEE Communications Letters*, 9(4), 2005.

[71] R. Sinha, C. Papadopoulos, and J. Heidemann. Internet packet size distributions: Some observations. Technical Report ISI-TR-2007-643, USC/Information Sciences Institute, May 2007. Orignally released October 2005 as web page *http://netweb.usc.edu/ rsinha/pkt-sizes/*.

[72] S. M. Stigler. The asymptotic distribution of the trimmed mean. *The Annals of Statistics*, 1(3):pp. 472–477, 1973.

[73] H. Sun, J. C.S. Lui, and D. K.Y. Yau. Defending against low-rate TCP attacks: Dynamic detection and protection. In *Proceedings of ICNP*, pages 196–205, 2004.

[74] G. Thatte, U. Mitra, and J. Heidemann. Detection of low-rate attacks in computer networks. In *Proceedings of the 11th IEEE Global Internet Symposium*, pages 1–6, Phoenix, Arizona, USA, April 2008. IEEE.

[75] G. Thatte, U. Mitra, and J. Heidemann. Parametric methods for anomaly detection in aggregate traffic. *ACM/IEEE Transactions on Networking*, page accepted to appear, August 2010. (Likely publication in 2011).

[76] G. Varenni, M. Baldi, L. Degioanni, and F. Risso. Optimizing packet capture on symmetric multiprocessing machines. In *Proceedings of 15th Symposium on Computer Architecture and High Performance Computing*, pages 108–115, Nov. 2003.

[77] P. Varga. Analyzing packet interarrival times distribution to detect network bottlenecks. In *EUNICE 2005: Networks and Applications Towards a Ubiquitously Connected World*, pages 17–29, 2006.

[78] T. Wolf, Y. Cai, P. Kelly, and W. Gong. Stochastic sampling for internet traffic measurement. In *IEEE Global Internet Symposium, 2007*, pages 31–36, May 2007.