Rating the Significance of Detected Network Events

A report
submitted in fulfillment
of the requirements for the degree
of
Master of Science
at
The University of Waikato

by

Meenakshee Mungro

THE UNIVERSITY OF
WAIKATO
Te Whare Wānanga o Waikato

Department of Computer Science
Hamilton, New Zealand
July 7, 2014

Abstract

Existing anomaly detection systems do not reliably produce accurate severity ratings for detected network events, which results in network operators wasting a large amount of time and effort in investigating false alarms. This project investigates the use of data fusion to combine evidence from multiple anomaly detection methods to produce a consistent and accurate representation of the severity of a network event. Four new detection methods were added to Netevmon, a network anomaly detection framework, and ground truth was collected from a latency training dataset to calculate the set of probabilities required for each of the five data fusion methods chosen for testing. The evaluation was performed against a second test dataset containing manually assigned severity scores for each event and the significance ratings produced by the fusion methods were compared against the assigned severity score to determine the accuracy of each data fusion method.

The results of the evaluation showed that none of the data fusion methods achieved a desirable level of accuracy for practical deployment. However, Dempster-Shafer was the most promising of the fusion methods investigated due to correctly classifying more significant events than the other methods, albeit with a slightly higher false alarm rate. We conclude by suggesting some possible options for improving the accuracy of Dempster-Shafer that could be investigated as part of future work.

Acknowledgements

I would like to acknowledge and thank the following people for their contributions to this project: Dr Richard Nelson, my supervisor, who has provided invaluable input throughout the year; Shane Alcock for keeping me motivated and helping debug my code; Brad Cowie, the WAND Systems Administrator, for keeping everything running smoothly and proofreading my thesis; Brendon Jones for also proofreading this thesis; members of WAND for the countless tea breaks and a tremendously pleasant experience throughout the duration of the project; friends and family, in particular my Dad, who provided moral support throughout the year.

A special thanks to my significant other, who was a neverending source of support and reminders to work throughout the year.

Contents

List of Figures

List of Acronyms

AMP	Active Measurement Project
API	Application Programming Interface
DoS	Denial-of-Service
FSN	Finite State Network
HMM	Hidden Markov Model
IDS	Intrusion Detection Systems
IP	Internet Protocol
NMS	Network Monitoring Systems
PCA	Principal Component Analysis
SVM	Support Vector Machine

Chapter 1

Introduction

Detecting anomalous behaviour in a network is important because it allows network operators to address problems on the network in a timely fashion, before the problems are noticed by the network users. Existing anomaly detection systems, such as Intrusion Detection Systems (IDS) and Network Monitoring Systems (NMS), tend to produce a high rate of false alarms, which results in network operators wasting time and energy in investigating erroneous events. Many IDS and NMS require extensive configuration and tuning of additional plugins for the system to behave as desired by the network operator, which can lead to misconfiguration issues and may not recognise every possible malfunction in the system.

Netevmon is a network event monitor, developed by the WAND Network Research Group [20], that runs a series of anomaly detection methods on network time series data. A monitor collects the measurements that form the time series and these measurements are used as input by Netevmon. The anomaly detection methods deployed within Netevmon produce severity scores for each detected event, but these scores are assigned arbitrarily and are often not an accurate indication of the true significance of the events.

We propose extending Netevmon to use data fusion to combine the evidence from the anomaly detection methods to form a single, accurate representation of the significance of an event that takes into consideration all of the evidence provided by the detection methods. We will focus exclusively on latency measurements, because many problems on the network are reflected in the changes in latency to other hosts and are thus clearly observable in a latency time series. Additionally, latency measurements were already available from the WAND Network Research Group's ongoing monitoring of latency using both a local instance of the Smokeping software [31] and measurements collected by monitors situated throughout New Zealand as part of the Active Measurement Project (AMP) [19]. This meant that a large volume of latency measurements was available to use for this work. If data fusion proves useful for latency time series, it will likely be useful for other network metrics as well, e.g. traffic volumes.

This project is not the first attempt at applying data fusion to network anomaly detection [28] [10] [42], but previous work has typically dealt with metrics extracted from passively captured network packets, rather than active latency measurements. However, the results from the previous work have shown that data fusion can be effective for network anomaly detection.

Data fusion requires evidence from multiple sources to be effective, but Netevmon only had two anomaly detection methods in use with latency time series at the time when this project was started. Hence, the first step was to investigate existing literature to find suitable anomaly detection methods for possible implementation within Netevmon to extend the number of potential sources of evidence. Of the candidates identified, three were chosen for implementation, namely T-Entropy [16], Hidden Markov Model (HMM) [23] and Real-time Changepoint Detection [4]. The implementation of these methods resulted in four new anomaly detection methods added to Netevmon, due to the T-Entropy method being applied to two separate metrics: one based on the standard deviation and the other based on the mean of the time series.

Each of the data fusion methods required a set of probabilities describing the likelihood that an event detected by a given anomaly detection method is significant. For this purpose, we collected ground truth data for 535 events detected from 25 Smokeping latency time series and manually assigned each event a severity score ranging from 0 to 5, with 0 being a false positive and 5 being a major event. Using this ground truth, the probabilities were derived by calculating the proportion of significant events, i.e. events with a severity score above 2 reported by each anomaly detection method. To determine whether the magnitude or the variability of the series should be accounted for when assigning the likelihood of significance, probabilities were also calculated for a set of "categories" defined by the mean latency and the variability of the time series at the time the event occurred. When considering the significance of an event, the data fusion algorithms would use the probabilities for the categories that described the current behaviour of the time series. Four categorisation methods (No categorisation, Latency only, Variability only, Latency and Variability) were implemented and compared in the course of this study.

To evaluate the accuracy of each data fusion method, a second dataset consisting of AMP-ICMP latency time series was collected and the events from the test dataset were manually processed in the same way as the Smokeping training data to derive ground truth. The events from the test dataset were run against the data fusion methods and the results reported by the fusion methods were compared with the manually assigned severity scores. Each fusion method reported a probability or belief that the event is significant. A threshold of 0.9 was used to determine when a data fusion method had identified that an event was significant, e.g. if the final probability reported by a fusion method was less than 0.9, the fusion method had not determined the event to be significant. An event with a severity score below 3 was correctly classified if the probability of significance never exceeded the threshold of 0.9. The opposite was true for significant events, i.e. events with a severity score of 3 and above must have exceeded the threshold to be correctly classified.

The results of the evaluation showed that none of the data fusion methods performed at a suitable level for practical deployment, primarily due to a high false negative rate, i.e. events marked as significant in the ground truth not exceeding the 0.9 threshold. The best performing method was Dempster-Shafer because it had the lowest rate of false negatives, despite achieving a slightly higher number of false positives than some of the other methods and therefore we conclude that future work should be focused on improving Dempster-Shafer to a satisfactory level of accuracy.

The contributions of this work can be summarised as follows:

- a ground truth latency time series dataset with manually assigned severity scores for known identified events has been collected that can be made available for distribution to other researchers;
- four new anomaly detection methods have been successfully implemented within Netevmon and are actively being used to detect network events by existing Netevmon deployments;
- five data fusion methods were implemented within Netevmon, namely

Dempster-Shafer, Averaging Belief Fusion, Cumulative Belief Fusion, Detector Count Heuristic, and Bayes' Theorem;

- the five data fusion methods have been evaluated against a test dataset and although we found that none of the methods achieved satisfactory results, Dempster-Shafer correctly identified the most significant events and would be therefore best suited for further experimentation;
- we discovered that the variability of a time series affects the reliability of the implemented detectors for latency time series, but found that categorising events based on the magnitude of the time series had little discernible benefit when deriving the probabilities for use with data fusion.

This thesis is organised as follows: Chapter 2 discusses network anomaly detection systems and the lack of anomaly detection systems that produce accurate severity ratings for detected events. It introduces our proposed solution to this problem: using data fusion to combine the evidence produced by the detectors deployed within Netevmon, a network event monitor. Chapter 3 describes the state of Netevmon at the start of this project, describes several anomaly detection methods that were potential detector candidates and discusses the selection of three anomaly detection methods (HMM [23], Real-time Changepoint Detection [4] and T-Entropy [16]) for implementation within Netevmon. Chapter 4 documents the implementation of the new detectors and the Symboliser, which is used to convert latency measurements to discrete characters for use by the Hidden Markov Model (HMM) and T-Entropy Detectors. Chapter 5 introduces the data fusion methods that will be evaluated and describes how they have been implemented within Netevmon. Chapter 6 describes the method for collecting ground truth from a set of Smokeping latency time series and the process of calculating the probabilities for each of the detectors. Chapter 7 documents the fusion method validation process, where each data fusion method was run against a second latency test dataset and their results compared against known ground truth for the test dataset. Chapter 8 presents and discusses the results of the fusion method validation and possible improvements that may increase the accuracy rate for some data fusion methods are identified. Finally, Chapter 9 concludes the report.

Chapter 2

Background

2.1 Network Anomaly Detection

Anomaly detection refers to the identification of abnormal behaviour that does not conform to the regular pattern in a dataset. In the case of a time series, an anomaly could be a sudden increase in value, a plunge or a change in the trend of the series. It is important to detect anomalies because they are often an indication that undesirable or unexpected behaviour is occurring and may warrant somebody's attention.

In the context of networking, anomaly detection refers to the identification of significant events or changes on a network. This is particularly useful for network operators since they are interested in knowing about problems on their networks as soon as possible. This allows the network operators to resolve the problems before the event escalates and affects the performance of the network, which could result in customer complaints.

There are several anomaly detection methods currently used by network operators to monitor real-time network traffic. Most of these methods belong to two broad categories: Intrusion Detection Systems (IDS) and Network Monitoring Systems (NMS).

An IDS is a software application or device that is used to detect malicious network traffic, e.g. malformed packets, network scans, Denial-of-Service (DoS) attacks, hacking attempts, etc. An IDS can be network based (NIDS), where a whole network is monitored for abnormal activity, or host based (HIDS), where individual hosts on the network are monitored. A NIDS analyses the traffic on a network and matches the packets to known network traffic attack signatures, while a HIDS monitors incoming and outgoing traffic for the host and detects whether the critical file system on a machine has been modified. A popular NIDS is Bro [32], an open-source passive network traffic analyser that monitors the network traffic on a link for any suspicious behaviour. Snort [37] is another widely used IDS that includes features such as real-time traffic analysis and packet logging, detecting a large range of probes and real-time alerting functionality.

A NMS continuously monitors a network to detect failing conditions (e.g. unreachable servers or congested links) and sends notifications when required to the network operator via email or text messages. An example of a widely used NMS is Nagios [17], which is a network monitoring tool that monitors many components of a network and alerts the network administrator when any of the components malfunction. Its basic functionality includes alerting, handling events and reporting events to network operators, but it can also be configured with a large variety of plug-ins to provide additional functionality, e.g. a visual front-end that allows visualising the state of the network or MySQL database statistics.

The downside of existing IDS and NMS software is that they can generate a large number of false alarms on a daily basis, which is undesirable because network operators either waste time and energy in investigating the false alarms or learn to ignore the system. While an NMS like Nagios can be tailored to the network operator's requirements, this often means spending a long time configuring multiple plug-ins and writing several scripts to ensure that the NMS will behave in the desired manner. Misconfiguration of the plug-ins and scripts is a common occurrence that means that the NMS will not function exactly as required by the network operator. It is also very difficult to tailor the configuration of an NMS to cater to every possible malfunction in the system.

Outside of IDS and NMS, network anomaly detection has been frequently investigated in literature, although most of the methods proposed are yet to see widespread use in industry. Shon et al. (2007) [41] proposed a novel Support Vector Machine (SVM) approach that uses unsupervised learning and provides a low false alarm rate. Thottan et al. (2003) [46] investigated a statistical signal processing method based on the detection of sudden changes on an Internet Protocol (IP) network and showed that it was an effective approach for detecting network anomalies on an IP network. Lu et al. (2009)

[27] described a method for detecting anomalies using a new network signal modelling approach with the combination of wavelet approximation and system identification theory. Taeshik et al. (2005) [40] investigated a machine learning based approach using a genetic algorithm for the selection of features and a Support Vector Machine (SVM) for classifying packets. Sekar et al. [38] attempted to achieve low rate of false alarms while detecting new attacks by using a new technique that integrates specification-based and anomaly-based intrusion detection. Krügel et al. (2002) [24] presented a technique that uses application specific knowledge of network services when extending network traffic models for the purposes of detecting suspicious data concealed in network packets. Androulidakis et al. (2009) [5] investigated the effect of intelligent flow sampling methods when detecting and classifying anomalies on a network. Ziviani et al. (2007) [49] proposed using non-extensive entropy, which is a one-parameter generalisation of Shannon entropy, in the detection of anomalies in an autonomous system (AS).

2.1.1 Netevmon

Netevmon is a software package developed by the WAND Network Research Group [20] which is designed to perform anomaly detection on a wide variety of different types of network time series data. As shown in Figure 2.1, Netevmon receives network time series data from the monitor that is collecting the measurements. The measurement must include a timestamp, a unique identifier for the time series that it belongs to, the method that was used to collect the measurement (e.g. Smokeping [31], traceroute, etc), and the observed value itself. The measurement method is used to determine which time series module should be used to analyse the time series. For example, if a Smokeping latency measurement is received, the Latency time series module will be used.

Within Netevmon, each anomaly detection method is implemented as a “Detector” module and each Time Series module defines a set of Detector modules that are suitable for finding anomalies in that particular type of time series data. A detector is not exclusive to a single Time Series module, e.g. the Plateau Detector in Figure 2.1 is used by both the Latency and Byte Count Time Series modules. Each time a new measurement is passed into a Time Series module, it is forwarded to an instance of each of the Detectors associated with the Time Series module. If the new measurement causes a Detector to decide that an event has occurred, an event object will be returned to the

Time Series module which will then be written into the Events database. This database can then form the back-end for a NMS-style dashboard which lists recent network events for an operator.

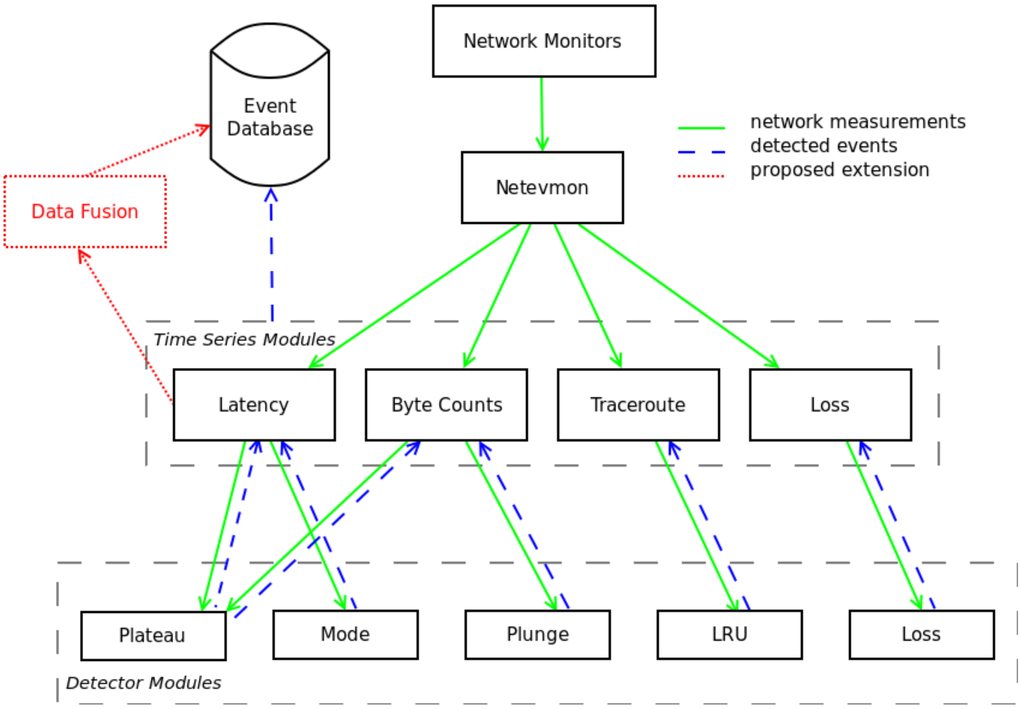

Figure 2.1: Architecture of Netevmon including the proposed Data Fusion component

Netevmon has a modular design which makes it easy to extend Netevmon to implement new anomaly detection methods as Detector modules or add support for new time series types, e.g. flow counts, packet counts, etc. New Detector modules can also be easily associated with existing time series modules. This makes Netevmon particularly suitable for developing and evaluating new anomaly detection methods, as the developer only needs to write code for the Detector module and add the new Detector to the desired Time Series module.

Currently, events detected by Netevmon are assigned severity scores (ranging from 0 to 100) by the Detector modules, but these scores are either arbitrary or assigned in a very simplistic fashion and are thus poorly representative of an event's true significance. For example, the Plateau Detector determines severity by dividing the average "normal" measurement by the average "anomalous" measurement. In this case, a latency change from 2ms to 4ms would have the same severity score as a change from 100ms to 200ms, but the latter case is likely to be of much more interest to the network operator than the former.

As a result, it is difficult for a network operator to evaluate the significance of the events currently detected by Netevmon.

Therefore, our proposed approach is to extend Netevmon to allow the Latency time series module to send detected events to a Data Fusion component. The Data Fusion component will then take into consideration all of the evidence produced by all of the Detectors and generate a final probability of significance for any given event. Although in Figure 2.1 only the Latency time series module is exporting events to the Data Fusion component, we anticipate that the other modules could also make use of data fusion.

We chose to focus exclusively on latency time series data because one of the more obvious indicators of a network experiencing problems is changes in the latency between hosts. For example, a failure in a router could result in packets using a different path or being unable to reach the destination, which might result in an increase in latency or missing latency measurements respectively. Latency can also be used to infer the location of a destination based on the mean latency; destinations in the same town will normally have a very low latency, whereas destinations on the other side of the globe will have a much higher latency. This feature is useful for network operators because it allows them to identify when the nearest server is no longer being used by observing an increase in latency. This is particularly significant for New Zealand operators, as domestic transit is cheap and high-bandwidth whereas international transit is expensive. Any change from domestic to international transit for a major content provider would demand immediate attention from the operator.

Another reason for choosing latency was that we had two ongoing sources of latency measurements that were readily available and supported by Netevmon. The first of these was the Smokeping [31] measurements collected by the WAND Network Research Group's internal monitoring, which measure latency from the WAND network to a variety of external targets. The second is the ICMP [33] test results collected by AMP [19] monitors located throughout New Zealand, which measure latency to both each other and to popular content and service providers, e.g. Google, DNS root servers, Twitter. This meant that we had a large number of latency time series that we could use as training and evaluation data for the purpose of this study.

2.2 Data Fusion

Network monitoring typically collects measurements at a regular interval, which creates a time series that can be used as input for an anomaly detection method. In literature, there are many anomaly detection methods for use with time series data, such as the Plateau Algorithm [29], Hidden Markov Model [23], Bayesian online changepoint detection [4], and an optimized K-nearest neighbors algorithm [34]. A more exhaustive list of anomaly detection methods can be found in [9].

While these methods can be effective at detecting changes in a time series, not all of the anomalies reported by each method is worthy of a network operator's attention. An ideal system would only alert an operator to significant anomalies, rather than alerting for every minor change observed on the network.

One potential solution is to utilise data fusion to combine the results from multiple anomaly detection methods to determine the significance of a detected network anomaly. Data fusion is defined as the combination of evidence from several sources to produce a unified result that is an accurate and consistent representation of the original evidence. There are several different methods for the combination of evidence (e.g. Dempster-Shafer [39], Averaging Belief Fusion [22], Bayes' Theorem [6], etc) which have been used in a wide variety of domains. One of the uses of data fusion in the weather monitoring domain is described in [26] where Dempster-Shafer is used to combine evidence from multiple radiation sensors for the purposes of predicting the presence of clouds. Data fusion also has military applications, for example in [15], Dempster-Shafer and fuzzy logic are used to aggregate the results of multiple sensors for the purposes of enhancing the performance and reliability of automatic target recognition systems. Data fusion is also used in semantic categorisation of images [13], where attributes of images such as colour, edge properties and texture, are combined with Bayes' Theorem to improve the performance of image categorisation.

In the context of network anomaly detection, each anomaly detection method acts as a source of evidence, either in favour of or against a detected anomaly being significant. The data fusion techniques combine the evidence from the anomaly detection methods to produce a final significance rating that takes into consideration all of the provided evidence.

Data fusion allows network operators to employ a wider range of anomaly detectors, without having to be concerned about the false positive rate of individual detectors. If a detector is known to be unreliable, any events it detects would require additional evidence before a network operator is alerted. Previous studies [28] [10] have shown that data fusion within the context of network anomaly detection is more effective than using the evidence of anomaly detection methods on their own. Other studies have shown that data fusion can provide an increase in the rate of detection while lowering the false alarm rate when combining results from different detection techniques [8] [47]. Some data fusion methods can also account for missing or unreliable evidence, which means that it is possible to obtain a significance rating without having to wait for the results of every detection method. This is important for providing timely feedback to network operators, as some detection methods are quicker to react to changes in the underlying time series than others.

2.2.1 Existing Work

At present, data fusion in the context of network anomaly detection has not been widely examined. There are only a few studies in literature where data fusion has been used to combine evidence from anomaly detection methods, such as [28], [10], [42].

Löf (2013) [28] proposed a method for evaluating network anomaly detection methods by using a novel methodology that uses a data fusion based approach. The new methodology is based on a new technique for creating anonymised network traces that include more information than other traces, while preserving the network users' privacy. A network trace annotation format that includes the information about the results of the anomaly detection methods is used, which allows keeping the implementation of the anomaly detection techniques private. Data fusion is then utilised to combine the output of multiple anomaly detection approaches for the purposes of identifying a wider range of network events. Five anomaly detection methods were chosen for testing, two of which used a time series as input (Holt Winters Forecasting [7] and Adaptive Single Exponential Smoothing Forecasting [48]). The three remaining anomaly detection methods, Snort [37], T-Entropy [16] and Bro [32] required network traces as input. The output of the anomaly detection methods were combined with two data fusion techniques, namely Dempster-Shafer and Averaging Belief Fusion. A third technique, "Majority Voting", was also used, where each anomaly detection method votes as to whether the detected event is an anomaly or not. The results of testing the performance of each individual anomaly detection method on their own and the three Data Fusion methods showed that Dempster-Shafer produced the best results.

Chatzigiannakis et al. (2007) [10] investigated the detection of anomalies in large scale networks using a data fusion based approach on a wide variety of sensors. The authors discussed a number of classifications of anomaly detection methods based on the data fusion approach relevant to each method, and chose to further study and test the performance of two different anomaly detection techniques: a Dempster-Shafer Theory of Evidence based method and a Principal Component Analysis (PCA) [21] based method. The two methods were evaluated using network traces captured on the network between the National Technical University of Athens (NTUA) and the Greek Research and Technology Network (GRNET) as input. The Dempster-Shafer method was applied to the sensor measurements using three metrics, ICMP packets out/in ratio, UDP packets in/out ratio, and TCP-SYN in/TCP-FIN out ratio. The measurements were then converted to basic probability assignments by using multiple thresholds per sensor measurement. The PCA-based approach was implemented using an algorithm based on M3L [11] and uses properties of the network trace as metrics, e.g. ICMP packets in, ICMP packets out, TCP flows in, TCP flows out, etc. The PCA-based method also used a training dataset that was captured on the same NTUA to GRNET network to create the network model. The results of the tests with the two data fusion based methods showed that the two methods could be used effectively to complement each other and increase the range of detected network anomalies.

Siaterlis and Maglaris (2004) [42] employed a Dempster-Shafer Theory of Evidence based approach to develop a new Denial-of-Service (DoS) anomaly detection method. Two sources of sensor input are used for the DoS detection method: simple traffic statistics from packet data (using libpcap [3]) captured using a preprocessor plug-in for Snort [37] (e.g. incoming and outgoing TCP SYN, TCP FIN, TCP, ICMP, UDP packet rates, etc) and data collected from an SNMP [1] collector and analyser (e.g. number of flow learn failures, queue drop counters and active flows). The state of the network is expressed as basic probability assignments by the sensors translating the measurements received as input. The basic probability assignments are transferred to the Dempster-Shafer fusion node by first converting the probability information to XML format and then using an extended version of the IDS communication protocol, called IDMEF [2]. The Dempster-Shafer fusion method then calculates the belief and plausibility values, and an averaging function is used in filtering short changes. The DoS detection engine was tested on a Gigabit Ethernet link on an academic research network that contained a combination of standard, online games, peer-to-peer and streaming audio traffic. The authors found evidence that showed that an attack would still be identified even when a sensor failed to detect the attack by combining information obtained from the other sensors.

Because the approaches mentioned in literature, [28], [10] and [42] require passive packet processing, they are not suitable for this project which is primarily focused on actively measured latency time series. Instead, we will implement and compare several data fusion techniques described in literature (e.g. Dempster-Shafer, Bayes' Theorem, Cumulative Belief Fusion [22]) to determine which one works best for network latency time series. An ideal data fusion method would correctly classify major network events as significant (minimising false negatives) while classifying a minimal number of insignificant events as significant (minimising false positives) to ensure that the network operator is only alerted to events that demand their attention.

Chapter 3

Detector Research

Data fusion requires several sources of evidence to be effective. In the case of Netevmon, each individual detector acts as a source of evidence. However, the number of detectors deployed within Netevmon at the start of this project was insufficient to be able to fairly evaluate the effectiveness of fusion methods. The implementation of new detectors within Netevmon was therefore necessary before belief fusion could be used effectively to combine the results reported by the detectors.

This chapter describes the existing detectors deployed within Netevmon at the time the project began, followed by a brief description of a number of potential detector candidates. Three methods are chosen for implementation and the reasons for choosing those detectors are described.

3.1 Existing Latency Time Series Detectors

At the start of this project, there were four detectors that used latency time series implemented within Netevmon:

Mode Detector

The Mode Detector has been designed to detect changes in the most common value observed by latency time series. For many paths, the latency is relatively constant and the Mode Detector can be effective at finding changes in the series.

The Mode Detector maintains a history buffer which stores the most recent 25 datapoints observed in chronological order and a frequency map which records how often each latency value appears in the history buffer.

The primary and secondary modes, which are the most frequent and second most frequent latency values respectively, are found by analysing the frequency map and the following conditions are evaluated to determine if a mode event has occurred:

- the history buffer must be full, since an accurate mode cannot be identified from only a few measurements;
- a previous mode is required for a change to be detected;
- the frequency of the primary mode must be greater than the minimum frequency threshold of 12;
- a mode change can only be reported if the most recent datapoint in the history buffer matches the primary mode. This is to avoid situations where old datapoints matching the previous primary mode are expired from the history buffer and frequency map, which would otherwise trigger a mode change;
- the difference between the frequency of the primary and secondary modes must be more than 5;
- the difference between the new and previous primary modes must be more than 3ms, as any latency change smaller than 3ms is unlikely to be worth alerting on;
- the previous mode must still be in the frequency map. If not, the series has not changed from one mode to another but has instead become constant again after a period of variability.

If all of these conditions have been met, the severity score of the event is obtained by calculating the ratio of the previous mode to the current one. Finally, the event is reported and the previous mode is set to the current primary mode.

Plateau Detector

The Plateau Detector is used to detect anomalous changes in the levels of the mean of a time series and is a modified implementation of the algorithm described in [29]. It maintains two buffers - a history buffer of measurements from the past 6 hours and a trigger buffer of 12 measurements. Initially, all new values are added to the history buffer. Once the history buffer is full, new measurements are evaluated by checking whether they fall within 3 standard deviations from the mean of the values currently in the history buffer. If the new measurement falls within the acceptable range, i.e. a non-anomalous measurement is seen, it is added to the history buffer and the oldest measurement is deleted, after which the statistics are updated. In this case, the oldest value in the trigger buffer is also removed if the trigger buffer contains values. If the new measurement falls outside of the acceptable range, the measurement is added to the trigger buffer. If the trigger buffer becomes full, an event is reported, provided the difference between the means of the two buffers is sufficiently large relative to the the history buffer mean.

Once it has been determined that there is an event at the current timestamp, the severity score is calculated depending on the value of the history and trigger buffer means. If one of the means is 0, the severity score is the other buffer mean. Otherwise, the severity score is the value of the larger mean divided by the smaller mean.

Loss Detector

Latency measurements also report whether packets were lost during the test. Normally, this value should be zero and any amount of lost packets may indicate a problem on the network. The Loss Detector maintains a history buffer of lost packet counts from the past 90 minutes. Each new measurement is added to the history buffer, after which the percentage of loss is calculated using the number of non-zero values in the buffer.

The Loss Detector is designed to detect two types of events:

- the percentage of non-zero values in the buffer exceeds a given threshold
- the number of consecutive non-zero values is greater than 3

The loss percentage is used to determine the severity level of the event, which is one of the following:

basic

A basic loss event occurs either when the loss percentage simply exceeds 33% or when the number of consecutive non-zero measurements exceeds 3;

escalated

An escalated event occurs when the percentage of loss in the buffer exceeds 66%;

extreme

An extreme loss event occurs when the history buffer is full of non-zero measurements.

When a loss event has been identified, the severity score is assigned depending on the current state of loss. If more than 3 non-zero measurements were found in the buffer, the severity score is 40; if the loss percentage exceeded 33%, the score is 60; escalated loss events have a score of 80 and finally, extreme loss events have a score of 100.

Latency TS Detector

The Latency TS Detector continuously monitors changes in the variability of a latency time series and is used to detect when the time series changes from constant to noisy or vice versa. In that case, it triggers a special "Noisy to Constant" or "Constant to Noisy" event with a severity score of 50. This component of Netevmon has the sole responsibility of informing the other active detectors of the current state of the time series, as they may behave differently based on the variability of the time series, e.g. using a larger threshold for anomalous behaviour if the series is noisy.

3.2 Potential Detector Candidates

Because it was not feasible to consider an exhaustive list of all the possible detectors that have been developed by other authors, we chose the methods presented in [9] as a suitable sample of possible options to investigate. Out of the anomaly detection methods discussed in [9], we chose methods that were likely to perform adequately with measurements from a latency time series. We also chose to investigate other methods based on their popularity in the context of anomaly detection, e.g. Holt-Winters Forecasting. The list of detectors that we evaluated and considered for implementation within Netevmon are:

Hidden Markov Model (HMM)

A HMM can be defined as a stochastic Finite State Network (FSN) that uses probabilistic rules to build an internal and external symbol sequence, which is then used to evaluate the likelihood of belonging to a previously determined model [35]. The HMM-based application proposed by [23] differentiates between regular and anomalous network traffic by using a predictive model.

First, the training stage builds a model using the training data. This includes the initialisation of parameters required for the model and the selection of a discrete symbol space for the conversion of states to an appropriate observation symbol. The Baum - Welch method [14] is used in estimating the parameters necessary for the HMM. The training model is iteratively evaluated and re-estimated until the required limiting point is reached. After the training session, the recognition phase allows testing if the network traffic is anomalous or normal. The anomaly detection system is provided discrete observation values and the recognition phase assigns a winning model (normal or abnormal) to the network traffic.

T-Entropy

T-entropy is a measure of complexity of any given string of symbols, where strings with more variability have higher T-entropy [16]. In [44], the average T-entropy rate of a stream of network packets is obtained by first converting the network packet features into a string. The stream data is simplified by mapping the contents of the IP packets into predetermined symbols to produce a string of characters that model the complexity of the original data. The T-complexity of the string is calculated using a set of values returned by an algorithm called T-decomposition. T-information, which is a linear measure for the amount of information in a string, is calculated from the resulting T-complexity value using an inverse logarithmic integral. Finally, T-entropy is obtained by calculating the gradient of T-information and is measured in natural information units. Events are then represented as changes in the rate of the average T-entropy.

While this approach used features of IP packets (e.g. the protocol field from the IP header), it could also be applied to a time-series of network measurements such as latency. This would require mapping the latency measurements to a discrete symbol space but would otherwise operate in a similar fashion to the original method.

Real-time Changepoint Detection

Changepoints are defined as sudden changes in the model of a sequence of data. There are a number of methods in literature that attempt to identify changepoints in time-series data [25] [4] [43] [45] [30] [12]. However, [43], [45] and [12] were ruled out as their approaches were not suitable for real-time processing. Of the methods that could be implemented for online changepoint detection, [4] was chosen as a candidate primarily because it explained the algorithm clearly in the literature.

Adams and MacKay (2007) [4] proposed a method where the parameters of the model are independent both before and after the changepoint. This method focuses on using causal predictive filtering, where a distribution of the next datum in the sequence is generated using previously-observed data. The observation sequence is assumed to be divisible into a series of product partitions that do not overlap based on the likelihood that conservative values belong to the same sequence. The delimitions between the resulting partitions represent the changepoints in the series.

Plunge Detector

The Plunge Detector is based on heuristics developed by the original authors of Netevmon. It is currently deployed on metrics other than latency, such as byte counts and user counts. It is used to detect sharp and sudden drops in time series that measure traffic volumes or other network utilisation metrics by using the median of previous measurements to represent the typical usage pattern. In this case, the mean is inappropriate since a downward trend at the start of the plunge will decrease the mean, meaning that a change would often not be detected. Each new measurement is added to the history buffer and once the median is calculated, the range is obtained by calculating the difference between the largest and smallest datapoints. A plunge event is reported if the following conditions are met:

- the datapoint is smaller than the smallest value currently in the history buffer;
- the difference between the median and the observed datapoint must exceed a minimum threshold (2 KB in the case of byte counts, 2 users in the case of user counts);

- the median of the values in the history buffer is at least 5ms, because plunges from very small values are less important to network operators.

k-Nearest Neighbours Algorithm

This algorithm is used in determining if a data value lies in the feature space by calculating the sum of distances to the k-nearest neighbours of the data value, which is called a kNN score for a data value. The authors of [34] propose a system that defines and categorises normality while also identifying new types of network attacks without any preliminary information of their existence. However, computing the kNN score to determine if an instance fits in with the rest of the sample is computationally expensive, especially when the complexity and volume of the network data is considered. Hence, the proposed method improves the efficiency of the algorithm by using a method similar to canopy clustering, which involves breaking down the feature set into smaller subsets. This eliminates the need to check every data point.

Exponential Smoothing with an Adaptive Response Rate

As stated by [48], forecasting systems that use exponential smoothing assume that the time-series data stems from basic processes that depend only on time, but that are laden with random noise. However, these simplistic systems do not react in a timely fashion to a sudden change in the time-series data. The authors of [48] proposed a forecasting system that uses exponential smoothing where the response rate is varied according to a tracking signal that is based on the amount of error observed in the previous forecasts. This allows the forecast method to adapt more quickly to significant changes in the underlying time-series, which is particularly relevant in the case of latency measurements because changes in latency tend to be relatively large. In simple cases, the authors suggest calculating the tracking signal by dividing the smoothed error by the smoothed absolute error, but more sophisticated methods could also be used if required.

Holt-Winters

The Holt-Winters [7] forecasting method is similar to the Exponential Smoothing with an Adaptive Response Rate method, but differs in that it can handle seasonal variations and can predict more than a single step in the future of a time series.

Holt-Winters' ability to handle seasonal data is especially useful in the context of latency time series since the forecast can therefore account for variation due to the network having varying loads at different times during the day. For example, latency may be higher or more variable during the peak hours of network usage.

Additionally, Holt-Winters deals with long term trends without using any adaptive methods. Holt-Winters forecasting states that the underlying time series model can be split into three major components: a baseline, a linear trend, and a seasonal effect. The algorithm used by the method applies exponential smoothing to gradually update the components since Holt-Winters Forecasting assumes that each component will evolve over a time period.

3.3 Chosen Candidates to Implement

In order to choose the most appropriate detectors to add to Netevmon, a list of requirements was drafted, namely:

- the detector must be easy to implement since the time allocated for detector implementation was limited;
- the detector must not require labelled training datasets for their training stage;
- the detector must not be computationally expensive since the detectors must be able to work in real-time and report events in a timely fashion;
- the detector should use a noticeably different approach from the existing detectors in order to detect different types of events;
- the detector does not require a perfect event detection rate since the data fusion method will ideally filter out false positives produced by unreliable detectors.

After studying the potential candidates and considering the set of requirements, T-Entropy, Hidden Markov Model (HMM) and Changepoint were selected. The chosen methods are not computationally expensive, which means that the techniques are appropriate for real-time analysis of network data.

They are all unsupervised methods, which means that they do not require labelled training datasets for their training stage. The algorithms used by the chosen methods were also clearly explained in the literature, which simplified the implementation. This was important given that there was a limited amount of time available for implementing new detectors. Additionally, they all differ in nature and this is necessary to detect a broader spectrum of anomaly types. Using the T-Entropy anomaly detection method was also advantageous because working code to convert a string of characters into an average T-entropy value was already available online [36] under the Apache License, which meant that the code would only need to be adapted to work within Netevmon. However, this method would require modifying Netevmon such that new measurements would be converted to discrete characters to form a string, which will in turn be used as input to the T-Entropy calculator. These discrete character strings will also be used as input by HMM, which will enable HMM to be implemented more easily.

The existing Plunge Detector was ruled out for implementation since it was designed primarily to monitor traffic volumes and user counts, where a sudden plunge to near zero would indicate a major network failure. However, latency measurements differ in that latency increases are generally more interesting events than plunges to network operators. Also, plunges in latency to near zero are very uncommon and do not typically indicate network failure. Network operators are instead more concerned about spikes in latency and thus, modifying the Plunge Detector to suit latency time series did not appear to offer much benefit over the candidates that were chosen.

Holt-Winters had been previously implemented within Netevmon and was found to produce inadequate results. In particular, the predictions made by the model would react too quickly to changes in the measurements and imitate the time series with a slight delay. Hence, it was difficult to detect when a measurement was anomalous since the predictions would learn the anomalous behaviour before a detection method would infer a change had occurred.

A prototype of Exponential Smoothing with an Adaptive Response Rate was developed but testing found that it had similar problems to the earlier Holt-Winters implementation, so this method was not pursued any further.

Chapter 4

Detector Implementation

Having identified suitable anomaly detection methods, the next phase of this research is the implementation of the new methods. Since the focus of this work involved investigating the combination of results from different sources, the time allocated for implementing new detectors was limited and the development of two of the three chosen detectors was assigned to other students.

This chapter primarily focuses on the implementation of the T-Entropy Detector. It also briefly describes the implementation of the Hidden Markov Model (HMM) and Changepoint detectors.

4.1 Implementation of the Symboliser

Since two of the proposed detectors required the conversion of latency measurements into a discrete symbol space, the first task involved developing algorithms for this purpose. The result of this work was the Symboliser.

The main objective of the Symboliser is to convert latency measurements into discrete characters. However, the range of possible latencies was too large to allow for a one-to-one mapping of latency measurements to symbols. One approach that we considered was to assign characters to ranges of latency measurements. For example, a measurement within the range of 0 to 5ms would be assigned an ‘a’ character, another measurement within the range of 6 to 10ms would be assigned a ‘b’ character, etc. This was ruled out because choosing appropriate ranges is difficult since a change from 5 to 10ms is much more significant than a change from 200 to 205ms. Also, there is the problem of two measurements that fall just either side of the range delimiter, e.g. with ranges of 5-10 and 10-15, a change from 9ms to 11ms would appear to have more impact than a change from 11ms to 15ms.

A better approach is to use metrics that measure the relative amount of difference between a latency measurement and the previously observed values, rather than any absolute difference or value. Relative metrics were also better suited for this work since they have a smaller range of pertinent values. This necessitated storing a history buffer of previous measurements since calculating the symbol metrics requires comparing the current measurement against statistics about previous measurements. The Symboliser currently maintains a history buffer of the past 3 hours of measurements, but the buffer can be configured to store varying amounts of data.

Before we can explain the Symboliser in more detail, we must first establish the terminology that will be used in this chapter. A *measurement* refers to the original latency value. A *symbol metric* refers to the result of applying one of the Symboliser metrics to a new measurement. A *symbol* refers to the character that a given symbol metric maps to. Finally, a *symbol string* refers to an ordered list of the symbols generated for the most recent measurements.

The Symboliser converts new measurements into a symbol metric, which is in turn mapped to a character from the discrete symbol space. The new character is then appended to the end of the symbol string for the symbol metric. Each new measurement observed will result in a new character getting added to the symbol string for the corresponding metric. Three metrics are supported by the Symboliser, namely:

diffStdDev

For a measurement x at position n in a time series, the diffStdDev metric is calculated as the number of standard deviations x is away from the mean of the history buffer. This metric can be expressed using the formula:

$$diffStdDev(x_n) = \frac{x_n - mean(H_{n-1})}{stddev(H_{n-1})} \tag{4.1}$$

where $H_{n\text{-}1}$ is the history buffer containing recent measurements up to and including $x_{n\text{-}1}$;

relMeanDiff

This symbol metric represents the difference between the current and previous measurements x_n and $x_{n\text{-}1}$, relative to the the mean of the history buffer. It is useful for detecting large changes in consecutive measurements and uses the formula:

$$relMeanDiff(x_n) = \frac{x_n - x_{n-1}}{mean(H_{n-1})} \tag{4.2}$$

propDiffMean

The propDiffMean symbol metric is calculated as the mean of the history buffer after adding the new measurement divided by the mean of the history buffer before the addition of the new value. It is represented with the following formula:

$$propDiffMean(x_n) = \frac{mean(H_n)}{mean(H_{n-1})} \tag{4.3}$$

Once the value for a symbol metric has been calculated, the next step is to convert that value into a discrete symbol. In the Symboliser, the symbol map consists of the default character '?', 'a' to 'z' for positive ranges, and 'A' to 'Z' for negative ranges. A range delimiter defines the range of values that correspond to each character in the symbol map. Assuming a range delimiter of 0.4, a symbol metric between -0.4 and 0.4 would result in a '?' character, which indicates a relatively normal measurement. Otherwise, the value is divided by the range delimiter and the quotient is used as an index into the symbol alphabet. The maximum index is 26 and any index larger than 26 is reduced to 26, i.e. a 'z' or 'Z' character depending on whether the symbol metric is negative or positive. For example, a symbol metric value of 1.3 and a range delimiter of 0.4 would result in an index of 3, which corresponds to a 'c'. Similarly, a symbol metric of -2.4 would result in an index of 6, which then results in an 'F' character since the value was negative. A very large change that would result in a symbol metric of 14 would have an index of 35 when using a range delimiter of 0.4, but the maximum index of 26 would be used, resulting in a 'z' character. The resulting character is then added to the end of the corresponding symbol string.

The symbol string returned by the Symboliser only contains the most recent characters added to the symbol string. Careful choice of the symbol string length is important because the length can have a significant impact on the event detection method that is using the string as input. In the case of long strings, measurements remain in the symbol string for longer and therefore, can affect the event detection long after those measurements are relevant. If the string length is too short, a string containing anomalous behaviour may not be sufficiently different from one that represents normal behaviour. The Symboliser implementation thus allows for the string length to be configured by the detector that is using it.

When using the Symboliser with the T-Entropy Detector, we encountered the problem where large changes would generate the same amount of entropy as smaller changes, i.e. an ‘a’ is equivalent to a ’z’ as far as entropy is concerned. This is a problem because any detection method designed to alert on the large changes would also alert on the insignificant ones as well. To mitigate this, an option was added to the Symboliser to append additional characters to the symbol string whenever the symbol represented a large change. For example, small changes that result in ’?’, ’a’, ’b’, ’A or ’B’ only have the 1 character added to the string; characters from ’c’ to ’e’ and ’C to ’E’ result in 2 duplicate characters getting added to the string; characters from ’f’ to ’i’ and ’F’ to ’I’ result in 3 duplicate characters getting added and finally, the remaining characters are added 4 times to the string. This ensures that significant events are detected faster due to the larger magnitude of the change being better reflected in the character strings.

4.2 Implementation of the T-Entropy Detector

The T-Entropy Detector was the first of the chosen candidates to be implemented. Source code for a stand-alone tool called `t-codes` [36] that would convert a string of characters into an average T-entropy value was available online under the Apache license. However, the tool being stand-alone meant that there was no usable Application Programming Interface (API) or shared library to use within Netevmon. Using the code in its original format would require executing the stand-alone tool and parsing the resulting output which would have been computationally expensive. This is especially problematic when working with online data as any delays in the processing of measurements would mean that recent measurements would not be processed in a timely fashion.

Hence, the source code included with the `t-codes` tool had to be converted to fit within the Netevmon framework. This involved adding debug output to analyse the order of the function calls, the effects of each function, and the results produced to ensure that all of the necessary code was correctly re-implemented. Once the code had been implemented as a shared library within Netevmon, it was tested for accuracy by comparing its output with that of the original `t-codes` tool for the same symbol string.

Each of the symbol metrics supported by the Symboliser will produce a distinct symbol string for the same time series of latency measurements. Because the symbol strings can differ, the T-entropy for each symbol metric is evaluated independently. As a result, the T-Entropy Detector acts as three separate detectors:

- the TEntropy-StdDev Detector, which uses the diffStdDev symbol metric;
- the TEntropy-RelMeanDiff Detector, which uses the relMeanDiff symbol metric;
- the TEntropy-MeanDiff Detector, which uses the propDiffMean symbol metric.

Converting latency measurements into T-entropy values produces a new time series which models the variation in the original data. However, changes in the new time series still need to be detected. Hence, the T-entropy values for each measurement are passed into a separate instance of the Plateau Detector, which detects anomalous behaviour in the T-entropy time series.

Figure 4.1 shows an example of a significant event that started at around 10:37 on 9 May 2014. The event was detected by the Plateau Detector at 10:42:02. Table 4.1 shows the effects of the event on the entropy calculated from the diffStdDev symbol metric. The T-entropy of the network just before the event started was constant at 0.35 since there was no abnormal behaviour that would introduce non-default characters and increase the entropy. The first abnormal measurement is observed at 10:37:03 with an increase from 4400ms to 133656ms, which inserts 4 'z' characters at the tail of the string and increases the T-entropy from 0.35 to 0.55. The T-entropy is further increased with the following measurements, where multiple characters were inserted because the difference between the current measurements and standard deviation of the previous measurements was still quite large.

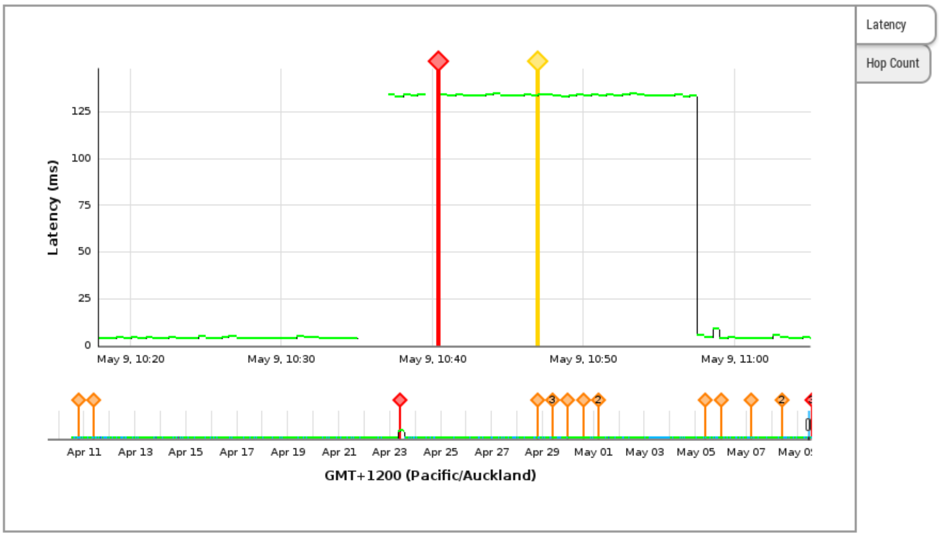

Figure 4.1: Screenshot of an event observed in a latency time series measured between prophet.cms.waikato.ac.nz and f.root-servers.net over IPv4

Time	Latency	String Representation	T-Entropy
10:34:03	3799	`??????????????????????????????`	0.35
10:34:31	3664	`??????????????????????????????`	0.35
10:35:02	4400	`??????????????????????????????`	0.35
10:37:03	133656	`??????????????????????????zzzz`	0.55
10:37:31	132746	`??????????????????????zzzzzzzz`	0.64
10:38:05	133631	`??????????????????zzzzzzzzzzzz`	0.68
10:38:33	133297	`??????????????zzzzzzzzzzzzzzzz`	0.69
10:39:00	133765	`??????????zzzzzzzzzzzzzzzzuuuu`	0.91
10:39:32	133773	`??????zzzzzzzzzzzzzzzzuuuupppp`	1.16
10:40:34	133677	`??zzzzzzzzzzzzzzzzuuuuppppmmmm`	1.33
10:41:00	133181	`zzzzzzzzzzzzzzuuuuppppmmmmjjjj`	1.42
10:41:32	133473	`zzzzzzzzzzzuuuuppppmmmmjjjjhhh`	1.62
10:42:02	133192	`zzzzzzzzuuuuppppmmmmjjjjhhhggg`	1.96
10:42:32	133289	`zzzzzuuuuppppmmmmjjjjhhhgggfff`	2.22

Table 4.1: Example of the T-Entropy during the event

4.3 Implementation of the Hidden Markov Model (HMM) Detector

The HMM detector was implemented by Andrew Mackintosh as a project for the University of Waikato Summer Research Scholarships. Similar to the T-Entropy Detector, it uses the Symboliser to generate a string of characters used to evaluate the model. However, it differs from the other detector in the configuration of the Symboliser, where the sample size and string size are set to half a day's worth of measurements, the range delimiter is 0.2 as opposed to 0.4, and the minimum difference between the current measurement and the previous mean is set to 4. It also disables the addition of multiple characters for larger symbol metric values and only uses a single metric: proportion of the current to previous mean, i.e. propDiffMean.

Before the HMM can begin detecting anomalous behaviour, it needs to be trained to determine the best model for the time series. During the training phase, every new measurement is added to the Symboliser until the default training size of half a day's worth of measurements has been observed. Then, a genetic algorithm is used to evaluate a large number of possible models for the training data and the best model is chosen. The genetic algorithm has been configured with a population size of 25 and is run for 500 generations.

Once the training model has been determined, the probability of the base case (string of 5 default '?' characters) is then calculated, after which the Symboliser is reset and the length of the string is set to 5 characters. The following measurements are then passed through the new Symboliser and the resulting string is tested against the training model to determine the probability of the string conforming to the training model. Because the probability values are often very small, they are scaled using the following formula: $scaled = \frac{max(-baseProb, -prob)}{min(-baseProb, -prob)}$, where $baseProb$ is the probability of the base case and $prob$ is the calculated probability for the current string.

Finally, the scaled probability is passed through a Plateau Detector, which detects any anomalous behaviour in the time series of the HMM probabilities. Similar to T-entropy, events in the original latency time series are then reported at the timestamp at which an anomaly is detected in the probability time series.

4.4 Implementation of the Changepoint Detector

The Changepoint Detector was implemented by Richard Sanger and is an implementation of the algorithm described in [4] with some modifications to suit the context of network anomaly detection. The Changepoint algorithm looks for sequences of consecutive measurements that are distinctly different from the values that comprise the preceding sequence. This is done by fitting a distribution to the sequence that is deemed to be “current” behaviour and using that distribution to determine the likelihood of recent measurements belonging to the current sequence. If the likelihood is low for several consecutive measurements, a changepoint is detected and an event is reported. The sequence that triggered the event then becomes the “current” sequence.

The distribution of regular measurements is based on a Gaussian distribution as opposed to the suggested student-t distribution because the Gaussian distribution was much simpler to implement and produced satisfactory results in testing. Another change was that a changepoint would only be detected if the sequence of anomalous measurements was no longer than 20% of the length of the sequence of previous normal behaviour. This ensures that the detector only reports recent changes as events by discarding events detected based on a change that occurred long ago.

Chapter 5

Data Fusion Methods

With the addition of new detectors within Netevmon, there are now sufficient sources of evidence available to investigate different data fusion methods. the data fusion methods will be used to integrate the output from each of the detectors into a final and accurate representation of the belief that an event is significant, i.e. the event is important enough for network operators to be interested in investigating it. Data fusion can also be used to calculate the likelihood that an event is a false positive because the detectors are not expected to be fully reliable and will occasionally report false positives. This is also useful to network operators by indicating which events may be false positives and should not be investigated.

In order to determine the best fusion method for integrating the output of several detectors using latency time series, five fusion methods were chosen as candidates for testing.

5.1 Dempster-Shafer Belief Fusion

The Dempster-Shafer Belief Theory [39] is a mathematical theory of evidence that is used to combine observations from several sources to produce a degree of belief which considers all of the presented evidence. It is based on the idea that degrees of belief for one question can be obtained from subjective probabilities of a related question, and Dempster-Shafer's rule is used to combine these degrees of belief when independent evidence is present. It also allows explicitly expressing uncertainty about subset probabilities, e.g. due to missing evidence in the results. This makes Dempster-Shafer's rule especially suited for Netevmon because the significance of some events may not be apparent at the point when the first detector fires, but may become less uncertain if more detectors later suggest that there is an event.

One commonly cited problem with Dempster-Shafer [18] is that it can produce counter-intuitive conclusions due to the final normalization step when conflicting sources are integrated because Dempster-Shafer ignores the weaker of the two opposing results. This is not an issue within Netevmon because detectors do not contradict the results of other detectors, but instead provide a cumulative proof of the possible presence of an event. As such, this problem can be safely disregarded in this context.

A key element of Dempster-Shafer is the concept of mass functions which, in this context, are pre-determined values that represent a prior belief about how likely it is that an event will be significant or a false positive when a particular detector fires. As will be explained in Chapter 6, the mass functions will be determined by examining the performance of each detector against a large sample of events where the significance has been manually classified.

The formula required for Dempster-Shafer's belief function is as follows:

$$bel(Sig) = (p(Sig).m(DetSig)) + (p(Any).m(DetSig)) + (m(DetSig).m(DetAny)) \tag{5.1}$$

$$bel(FP) = (p(FP).m(DetFP)) + (p(Any).m(DetFP)) + (p(FP).m(DetAny)) \tag{5.2}$$

$$bel(Any) = p(Any).m(DetAny) \tag{5.3}$$

where:

bel(Sig)

The belief that an event is significant;

bel(FP)

The belief that an event is a false positive;

bel(Any)

The belief that an event is neither significant or a false positive;

p(Sig)

The prior belief that the event is significant based on the detectors that have fired previously;

p(FP)
The prior belief that the event is a false positive based on the detectors that have fired previously;

p(Any)
The prior belief that the event is neither significant nor a false positive based on the detectors that have fired previously;

m(DetSig)
The degree of belief that an event reported by this detector is significant based on previous detector history;

m(DetFP)
The degree of belief that an event reported by this detector is a false positive based on previous detector history;

m(DetAny)
The degree of belief that an event reported by this detector is neither significant nor a false positive based on previous detector history.

5.2 Bayes' Theorem

Bayes' Theorem [6] is a mathematical formula used to calculate conditional probabilities. It can be used as a means of determining how the probability that a theory is true is affected by new evidence. Bayes' Theorem was chosen as a fusion method because it is a well-studied and widely used method in the field of statistical inference using probabilities. It is also useful within the context of event detection using belief fusion where the probability of an event being significant will be updated each time a new detector fires for any event group.

Implementing Bayes' Theorem within Netevmon will require computing the formula:

$$P(Sig|Det) = \frac{P(Det|Sig).P(Sig)}{(P(Det|Sig).P(Sig)) + (P(Det|!Sig).P(!Sig))} \tag{5.4}$$

$$P(FP|Det) = \frac{P(Det|FP).P(FP)}{(P(Det|FP).P(FP)) + (P(Det|!FP).P(!FP))} \tag{5.5}$$

which involves the following components:

P(Det)
: The probability that a particular detector has fired;

P(Sig)
: The probability that an event is significant;

P(FP)
: The probability that an event is a false positive;

P(!Sig)
: The probability that an event is not significant;

P(!FP)
: The probability that an event is not a false positive;

P(Det|Sig)
: The probability that the detector will fire if an event is significant;

P(Det|FP)
: The probability that the detector will fire if an event is a false positive;

P(Det|!Sig)
: The probability that the detector will fire if an event is not significant;

P(Det|!FP)
: The probability that the detector will fire if an event is not a false positive;

P(Sig|Det)
: The probability that an event is significant given that the detector has fired;

P(FP|Det)
: The probability that an event is a false positive given that the detector has fired.

5.3 Cumulative Belief Fusion

The authors of [22] have proposed Cumulative Belief Fusion as a solution to the contradictory evidence problem introduced by Dempster-Shafer's Belief Fusion. The cumulative rule is appropriate for scenarios where the accumulation of evidence is used to combine independent belief fusions, whereas Dempster-

Shafer's rule combines beliefs by normalised conjunction. The latter ignores the weaker of the conflicting evidence, while the proposed method attempts a fair and equal representation of the conflicting beliefs.

This method is especially appropriate for use within Netevmon since the detectors are independent and their results can be accumulated to determine whether an event is significant or not.

The formula to update the probability that an event is significant when a detector fires is:

$$P(Sig) = \frac{(p(Sig).P(DetAny)) + (P(DetSig).p(Any))}{(P(DetAny) + p(Any)) - (P(DetAny).p(Any))} \tag{5.6}$$

$$P(FP) = \frac{(p(FP).P(DetAny)) + (P(DetFP).p(Any))}{(P(DetAny) + p(Any)) - (P(DetAny).p(Any))} \tag{5.7}$$

$$P(Any) = 1 - (P(Sig) + P(FP)) \tag{5.8}$$

where p(Any) and P(DetAny) are both not equal to 0, and:

$$P(Sig) = (0.5.p(Sig)) + (0.5.P(DetSig)) \tag{5.9}$$

$$P(FP) = (0.5.p(FP)) + (0.5.P(DetFP)) \tag{5.10}$$

$$P(Any) = 1 - (P(Sig) + P(FP)) \tag{5.11}$$

where p(Any) and P(DetAny) are both equal to 0, and:

P(Sig)

The probability that an event is significant;

P(FP)

The probability that an event is a false positive;

P(Any)

The probability that an event is neither significant nor a false positive;

p(Sig)

The prior probability of the event being significant based on the detectors that have fired previously;

p(FP)

The prior probability of the event being a false positive based on the detectors that have fired previously;

p(Any)
: The prior probability of the event being neither significant nor a false positive;

P(DetSig)
: The probability that an event reported by this detector is significant;

P(DetFP)
: The probability that an event reported by this detector is a false positive;

P(DetAny)
: The probability that an event reported by this detector is neither significant nor a false positive;

The last three probabilities, i.e. P(DetSig), P(DetFP) and P(DetAny) are equivalent to the mass function values used by Dempster-Shafer.

5.4 Averaging Belief Fusion

Averaging belief fusion is the second method proposed in [22] and is a solution to Dempster-Shafer's problem for the same reasons as the Cumulative Belief Fusion method.

Averaging belief fusion is appropriate for scenarios using dependant belief functions, where combining belief functions is achieved by averaging the evidence. This means that this fusion method is not entirely suited for use within Netevmon since the detectors are fully independent and do not rely on the results of other detectors to reach a conclusion. However, because the averaging method is simple to implement, it was included as a candidate fusion method for this study primarily to serve as a point of comparison against the other fusion methods.

The averaging belief fusion is simply an average of the probabilities that an event is significant for each individual detector that fired. It uses the same probability values derived from the mass functions for Dempster-Shafer with the following formula:

$$P(Sig) = \frac{1}{n} \sum P(DetSig) \tag{5.12}$$

$$P(FP) = \frac{1}{n} \sum P(DetFP) \tag{5.13}$$

$$P(Any) = \frac{1}{n} \sum P(DetAny) \tag{5.14}$$

5.5 Detector Count Heuristic

This heuristics-based fusion method was designed as a simple fusion method that does not require any pre-computed probability values regarding the accuracy of the detectors that fired. It was suggested by the author of Netevmon based on his experiences with the events produced by the detectors in a production environment.

It is based on the principle that a significant event is much more likely to cause most, if not all, of the detectors to fire, but assumes that all detectors are equally reliable. If four or more detectors fire for any event group, the event is instantly classified as significant. If fewer detectors fire, the count of detectors that fired is multiplied by 0.2 to get the final probability of significance. The formula used for this method is:

$$P(Sig) = \begin{cases} 0.2x, & \text{if } x \leq 3 \\ 0.9, & \text{otherwise} \end{cases}$$

where:

P(Sig)

The probability that an event is significant;

x

The number of detectors that fired for an event group.

If this method performs well in our evaluation, it would suggest that probability-based belief fusion is unnecessary, which is especially convenient since the Detector Count Heuristic method is computationally inexpensive and easy to implement than the other methods.

Chapter 6

Ground Truth

The implemented data fusion methods require a set of probabilities for each of the detectors, e.g. the probability of a detector triggering given a significant event or false positive, the probability of an event in a time series being significant, etc. Due to the nature of the fusion methods, different probabilities were required for Bayes' Theorem and Dempster-Shafer, but the same probabilities can be used for Dempster-Shafer, Averaging and Cumulative Belief Fusion methods. Hence, a dataset of events with known severity ratings was collected for the purposes of calculating the different probabilities to use for each detector.

This chapter focuses on the detectors chosen for use with the data fusion methods, how the ground truth data was collected and assigned a severity rating, followed by the selection and calculation of the probability values for each individual detector.

6.1 Detector Selection

Before the ground truth data collection could begin, we needed to confirm which detectors would be used with the data fusion methods. The Loss Detector was ruled out because it has been implemented specifically for detecting loss in a time series, which is a different type of event from the latency changes that the other detectors monitor. The Latency TS Detector was also not chosen for this task since it had been designed to primarily monitor changes in the variability of the time series and only triggers "Noisy to Constant" or "Constant to Noisy" events, which are more useful for informing the other detectors of the current state of the time series.

Having implemented the three metrics (diffStdDev, relMeanDiff, and propDiffMean) for the TEntropy Detector, we obtained three new detectors: TEntropy-StdDev, TEntropy-MeanDiff and TEntropy-RelMeanDiff. After thoroughly testing and analysing the performance of the three detectors, we observed that the TEntropy-RelMeandiff Detector was too unreliable for event detection in its current form and it was not clear if it could be easily improved.

Therefore, the detectors that had been chosen for use with data fusion were:

- Plateau Detector
- Mode Detector
- TEntropy-StdDev Detector
- TEntropy-MeanDiff Detector
- Changepoint Detector
- Hidden Markov Model (HMM) Detector

6.2 Smokeping Ground Truth Data

Because Netevmon was primarily analysing Smokeping [31] measurements at the time when the processing of ground truth data was required, Smokeping data was used as the ground truth. The Smokeping data is a duplicate of the measurements collected by the real-time monitoring system used by the WAND group at the University of Waikato to monitor their connectivity to external sites. Smokeping was also the only data source available at the time that could be used to provide a sufficiently large sample of events across multiple latency time series.

Smokeping latency measurements are collected by sending batches of ICMP echo request packets out to the network and measuring the time taken for the packets to travel from the source to the destination and back again. The WAND Smokeping system sends 20 packets for each measurement every 5 minutes, which is a standard configuration for Smokeping. Afterwards, the resulting round trip times for the 20 packets are sorted and the median is selected as the final latency value used by Netevmon. When the measurements are graphed, the variation in the individual results for each burst of packets is represented by drawing the remaining values as lighter gray shades (known as "smoke") in the background. Lost packets are displayed by changing the line colour based on the loss percentage, as shown in Figure 6.1.

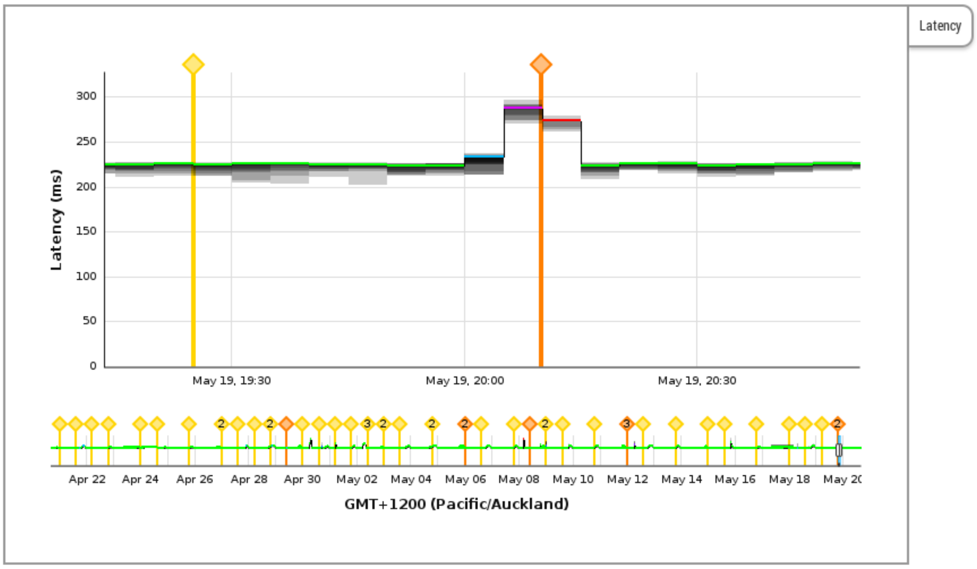

Figure 6.1: Screenshot of the Smokeping latency on the network between WAND and google.com. The vertical lines represent events detected by Netevmon.

To gather the ground truth data, Netevmon was run against a wide variety of Smokeping time series and each event reported by one of the six detectors was written to a file for later analysis. While we expected the detectors to trigger events whenever there was a change in the time series, there was also the possibility of false negatives, i.e. no detectors identifying irregular behaviour on the network when anomalous changes were happening. It is important to bear in mind that the aim of this research is to evaluate how well fusion methods work with the events that are being detected, rather than the effectiveness of the detectors themselves. After going through several weeks of data for 25 Smokeping time series and manually inspecting 535 event groups, there were no obvious false negatives.

Every event reported by at least one detector was manually examined to determine the starting time of the event. This is used primarily to analyse the delay between the event starting and the detector recognising that an event has occurred. The delay is important for determining how fast the individual detectors are at identifying significant events. Detectors that take a long time to identify an event are seldom useful to network operators, even if the detector itself is completely reliable. Any events that occur within close proximity of the event start time are manually grouped together as these are likely to be the same event. For each event group, the following information is recorded:

- the timestamp when the time series started changing, i.e. the start of the event; This is typically several measurements prior to the first detector firing;
- the severity score of the event group. This is a value ranging from 0 to 5, where 0 indicates a false positive, 1 indicates an insignificant event, 2 means that the event is neither significant nor insignificant (also termed "undecided"), and 3 - 5 indicate significant events of increasing importance;
- the timestamp at which each detector fired for an event group, or 0 if the detector did not fire;
- the delay in minutes between the start of the event and the time when the detector determined that there was an event, if the detector fired.

We assigned the severity scores to each event group by manually inspecting the state of the time series at the time when the event started and assigning a severity rating based on the magnitude of the change and the duration of the anomaly. While the severity ratings were subjective, a consistent logic was applied when classifying each events to ensure that the severity ratings would be consistent across the multiple time series. Also, when in doubt, we sought second opinions from colleagues to clarify any uncertainty.

Table 6.1 displays a summary of the events detected by each detector across 25 Smokeping latency time series. Each row contains the number of events for each severity score that were reported by each detector. The "Detected Events" row contains the number of unique events observed across all of the detectors, i.e. the maximum number of events observed with that severity score. Because an event can be reported by multiple detectors, the values in this row should not be interpreted as a sum of the other rows in the table.

Table 6.1 reveals some of the characteristics of each detector. For example, while the Mode Detector fired less often when compared to the other detectors, almost all of the events it detected were significant, suggesting that any event reported by the Mode Detector would be of interest to an operator. On the other hand, the detector that fired the most often, TEntropy-Stddev, has a high rate of detecting insignificant events, which implies that it is more prone to false alarms than the other detectors.

	Severity Score						
Detector	0	1	2	3	4	5	Total
Plateau	0	23	46	40	25	76	210
Changepoint	10	8	28	16	13	32	107
TEntropy-StdDev	6	44	80	63	30	82	305
TEntropy-MeanDiff	14	13	50	41	28	80	226
Mode	4	0	0	9	23	56	92
HMM	6	27	59	46	26	77	241
Detected Events	34	87	159	93	52	110	535

Table 6.1: Summary of the manually classified severity scores for the events detected by each detector

Table 6.2 shows the average delays between the start of an event and the timestamp at which each detector identified that an event occurred. Because the average interval between measurements is 5 minutes, the delays become quite large, especially for the Mode and Plateau detectors. Large delays are not ideal in a practical context because the delays prevent network operators from handling events in a timely fashion. Although the Mode Detector is a reliable indicator of significance, the delay between the event occurring and the detector alerting makes it difficult for network operators to react on time. By contrast, the T-Entropy detectors are less reliable than the Mode Detector, but are much quicker at detecting events, which means that network operators are alerted to potentially significant events much faster and are thus able to respond in a timely fashion.

Even though the average delay for the Mode Detector is 75.2 minutes, this corresponds to only 15 measurements. If measurements were taken more frequently, the average delay should decrease correspondingly.

	Severity Score					
Detector	1	2	3	4	5	All Events
Plateau	56.7	60.3	54	68.4	51.9	56.7
Changepoint	45	47.7	42	83	47.8	52
TEntropy-Stddev	33.8	31.9	30.9	33.3	30.6	31.8
TEntropy-MeanDiff	39.6	29.7	34.6	34	28.8	31.3
Mode	N/A	N/A	86.6	69.6	73.7	75.2
HMM	44.4	31.7	32.3	40.8	31.5	34.4
All Detectors	44.6	38	41	51.9	41.8	42.5

Table 6.2: Summary of the average delay in minutes before a detector fires for events that are not false positives

Before commencing the calculation of probability values for each detector, we decided to divide the latency time series into categories. This was important because we anticipated that individual detector performance would differ depending on the magnitude of the latency values and the noisiness of the measured time series. To account for this, we divided the time series in our sample into several categories based on their average latency and their variability. We also chose to create a "No Categorisation" category where there would be no splitting, i.e. a category that would include all of the latency time series in our sample, because it would allow us to compare the effects of the different categorisation methods on the final probabilities. The categorisation methods and the resulting categories are:

No Categorisation

This categorisation method only has one category containing all of the events, i.e. no splitting based on latency or variability;

Latency

This categorisation method separates the latency time series into different categories based on the mean latency of the time series, which are:

- 0 - 5ms (same or nearby cities)
- 5 - 25ms (within the rest of New Zealand)
- 25 - 100ms (Australia)

- 100 - 300ms (United State and Europe)
- 300+ms (rest of the world)

The latency categories were chosen because they are representative of the location of the target relative to a source located in New Zealand. Targets that are in close proximity of the source (e.g. in the same city) will typically have a much lower latency than targets that are much further away.

Variability

This categorisation method separates the latency time series into two categories based on their variability, which are:

- Constant
- Noisy

The variability of a stream is “Constant” when the majority of the measurements are within 2ms of a consistent value. A stream is classified as “Noisy” when the measurements tend to vary much more from measurement to measurement, rather than settling on a constant value.

Latency and Variability This categorisation method separates the latency time series into multiple categories based on both their variability and latency, e.g. 0 - 5ms & Noisy, 5-10ms & Noisy, etc.

Table 6.3 contains a list of the Smokeping latency time series used in the sample for the ground truth. The range of the mean latency, the variability, the number of false positives (assigned at severity score of ‘0’), insignificant events (severity score rating of ‘1’), events that are neither significant nor insignificant (severity score of ‘2’) and significant events (severity score of ‘3’ to ‘5’) is included.

As shown in the table, there are only 4 latency time series with a variability of “Noisy” because only a few time series qualified as “Noisy” in Smokeping tests. Some latency groups had no noisy time series, e.g. ‘100 - 300’ and ‘300 +’. Others, such as ‘0 - 5’, only had 1 “Noisy” time series, which is not ideal because the resulting sample is not diverse, and therefore biased towards the behaviour of one series. Because of the limited number of different time series collected by WAND’s Smokeping monitor, processing other series to increase the sample size of a specific category was not a viable option.

Time Series	IP Version	Mean Latency (ms)	Variability	FP Events	Significant Events	Insignificant Events	Undecided Events	Total Events
trademe.co.nz	IPv4	0 - 5	Noisy	15	4	7	8	34
htb.gtw.rurallink.co.nz	IPv4	0 - 5	Constant	0	2	13	8	23
ariel-248.waikato.ac.nz	IPv4	0 - 5	Constant	5	5	1	14	25
pegasus.waikato.ac.nz	IPv4	0 - 5	Constant	0	15	0	0	15
fx.net.nz	IPv4	0 - 5	Constant	0	4	0	7	11
fx.net.nz	IPv6	0 - 5	Constant	0	4	3	2	9
google.com	IPv4	5 - 25	Noisy	1	31	2	12	46
ps02-wlg.reannz.co.nz	IPv4	5 - 25	Constant	0	1	0	5	6
ps02-wlg.reannz.co.nz	IPv6	5 - 25	Constant	1	0	1	1	3
ns1.dns.net.nz	IPv4	5 - 25	Constant	1	4	0	4	9
ns1.dns.net.nz	IPv6	5 - 25	Constant	0	6	2	3	11
ns4.dns.net.nz	IPv4	5 - 25	Constant	0	11	1	6	18
youtube.com	IPv4	25 - 100	Noisy	1	41	3	1	46
apnic.net	IPv4	25 - 100	Noisy	0	4	0	5	9
ftp.monash.edu.au	IPv4	25 - 100	Constant	1	5	3	0	9
google.com	IPv4	25 - 100	Constant	0	14	0	2	16
8.8.8.8	IPv4	25 - 100	Constant	0	5	0	0	5
youtube.com	IPv6	100 - 300	Constant	0	13	1	1	15
netflix.com	IPv4	100 - 300	Constant	0	3	4	1	8
facebook.com	IPv4	100 - 300	Constant	0	5	1	3	9
facebook.com	IPv6	100 - 300	Constant	0	3	3	3	9
apnic.net	IPv4	100 - 300	Constant	2	23	2	21	48
arin.net	IPv4	100 - 300	Constant	0	3	7	4	14
afrinic.net	IPv4	300 +	Constant	7	11	2	13	33
lacnic.net	IPv4	300 +	Constant	0	34	31	35	100

Table 6.3: The number of false positive, significant, insignificant, and undecided events for each individual time series in the sample for our ground truth

The probability masses and values used for each individual detector are calculated as follows:

$$m(DetSig) = \frac{\text{Count of significant events}}{\text{Total events}} \tag{6.1}$$

$$m(DetFP) = \frac{\text{Count of false positive events}}{\text{Total events}} \tag{6.2}$$

$$m(DetAny) = \frac{\text{Count of events that are neither significant nor false positive}}{\text{Total events}} \tag{6.3}$$

$$P(Sig) = \frac{\text{Count of all significant events in a category}}{\text{Total events in a category}} \tag{6.4}$$

$$P(!Sig) = 1 - P(Sig) \tag{6.5}$$

$$P(FP) = \frac{\text{Count of all false positive events in a category}}{\text{Total events in a category}} \tag{6.6}$$

$$P(!FP) = 1 - P(FP) \tag{6.7}$$

$$P(Det|Sig) = \frac{\text{Count of significant events identified by a detector}}{\text{Total significant events in a category}} \tag{6.8}$$

$$P(Det|!Sig) = \frac{\text{Count of all non-significant events identified by a detector}}{\text{Total non-significant events in a category}} \tag{6.9}$$

$$P(Det|FP) = \frac{\text{Count of false positive events identified by a detector}}{\text{Total false positive events in a category}} \tag{6.10}$$

$$P(Det|!FP) = \frac{\text{Count of all non-false positive events identified by a detector}}{\text{Total non-false positive events in a category}} \tag{6.11}$$

Tables 6.4 and 6.5 contain the probability masses used for Dempster-Shafer and Tables 6.6 and 6.7 contain the probability values used for Bayes' Theorem for the TEntropy-StdDev and Mode Detectors respectively. The probability masses and values used for the other detectors can be found in the tables given in Appendices A and B. In cases where there were no samples for a category, we used the probabilities from the equivalent category where the latency is disregarded. Probability values where no sample was available are marked with an asterisk in the probability tables.

The "None" variability category is used whenever the categorisation method does not take variability into account, i.e. noisy and constant time series are considered equivalent. The "No Latency" latency category is the equivalent for categorisation methods that ignore the latency of a time series. The "No

Latency" - "None" row is used by the "No Categorisation" method, where both latency and variability have been disregarded.

Careful examination of the probability masses and values show that they accurately reflect the observed behaviour for each detector. For example, we observed that the Mode Detector has a very high chance of detecting a significant event, e.g. it has a probability mass of more than 0.9 for 14 out of the 18 categories. This is supported by the evidence in Table 6.1, where 88 out of the 92 Mode events were significant. By contrast, Table 6.1 showed that 43% of the events identified by the TEntropy-StdDev Detector were either false positives, insignificant or neither significant nor insignificant. This is supported by the probability masses and values being relatively lower for the TEntropy-StdDev Detector, e.g. 8 out of 18 categories have a probability mass of below 0.6, with no mass values above 0.9.

Latency (ms)	Variability	m(DetSig)	m(DetFP)	m(DetAny)
0 - 5	Constant	0.81	0.00	0.19
	Noisy	0.17	0.33	0.50
	None	0.58	0.12	0.30
5 - 25	Constant	0.82	0.00	0.18
	Noisy	0.71	0.00	0.29
	None	0.76	0.00	0.24
25 - 100	Constant	0.82	0.00	0.18
	Noisy	0.84	0.00	0.16
	None	0.83	0.00	0.17
100 - 300	Constant	0.47	0.02	0.51
	Noisy	0.88*	0.00*	0.12*
	None	0.47	0.02	0.51
300 +	Constant	0.40	0.00	0.60
	Noisy	0.88*	0.00*	0.12*
	None	0.40	0.00	0.60
No Latency	Constant	0.52	0.11	0.37
	Noisy	0.88	0.00	0.12
	None	0.57	0.09	0.34

Table 6.4: Probability masses used for the TEntropy-StdDev Detector for Dempster-Shafer, Averaging, and Cumulative Belief Fusions.
Probability masses marked with an asterisk use the probability samples for the equivalent No Latency category because no samples were available.

Latency (ms)	Variability	m(DetSig)	m(DetFP)	m(DetAny)
0 - 5	Constant	1.00	0.00	0.00
	Noisy	0.00	1.00	0.00
	None	0.79	0.21	0.00
5 - 25	Constant	1.00	0.00	0.00
	Noisy	1.00	0.00	0.00
	None	1.00	0.00	0.00
25 - 100	Constant	1.00	0.00	0.00
	Noisy	1.00	0.00	0.00
	None	1.00	0.00	0.00
100 - 300	Constant	0.80	0.07	0.13
	Noisy	0.92*	0.08*	0.00*
	None	0.87	0.07	0.07
300 +	Constant	1.00	0.00	0.00
	Noisy	0.92*	0.08*	0.00*
	None	1.00	0.00	0.00
No Latency	Constant	0.96	0.02	0.02
	Noisy	0.92	0.08	0.00
	None	0.95	0.04	0.01

Table 6.5: Probability masses used for the Mode Detector for Dempster-Shafer, Averaging, and Cumulative Belief Fusions.
Probability masses marked with an asterisk use the probability samples for the equivalent No Latency category because no samples were available.

Latency (ms)	Variability	P(Sig)	P(!Sig)	P(Det|Sig)	P(Det|!Sig)	P(FP)	P(!FP)	P(Det|FP)	P(Det|!FP)
0 - 5	Constant	0.36	0.67	0.57	0.08	0.06	0.94	0.00	0.27
	Noisy	0.12	0.88	0.50	0.33	0.44	0.56	0.27	0.42
	None	0.29	0.71	0.56	0.17	0.17	0.83	0.20	0.30
5 - 25	Constant	0.47	0.53	0.64	0.12	0.04	0.96	0.00	0.38
	Noisy	0.67	0.33	0.65	0.53	0.02	0.98	0.00	0.62
	None	0.57	0.43	0.64	0.28	0.03	0.97	0.00	0.50
25 - 100	Constant	0.80	0.20	0.75	0.67	0.03	0.97	0.00	0.76
	Noisy	0.82	0.18	0.69	0.60	0.02	0.98	0.00	0.69
	None	0.81	0.19	0.71	0.63	0.02	0.98	0.00	0.71
100 - 300	Constant	0.49	0.51	0.80	0.87	0.02	0.98	1.00	0.83
	Noisy	0.59*	0.41*	0.66*	0.44*	0.13*	0.87*	0.24*	0.62*
	None	0.49	0.51	0.80	0.87	0.02	0.98	1.00	0.83
300 +	Constant	0.34	0.66	0.73	0.56	0.05	0.95	0.00	0.65
	Noisy	0.59*	0.41*	0.66*	0.44*	0.13*	0.87*	0.24*	0.62*
	None	0.34	0.67	0.73	0.56	0.05	0.95	0.00	0.65
No Latency	Constant	0.43	0.57	0.71	0.47	0.04	0.96	0.12	0.60
	Noisy	0.59	0.41	0.66	0.44	0.13	0.87	0.24	0.62
	None	0.47	0.53	0.70	0.46	0.06	0.94	0.18	0.60

Table 6.6: Probability values used for the TEntropy-StdDev Detector for Bayes' Theorem. Probability masses marked with an asterisk use the probability samples for the equivalent No Latency category because no samples were available.

Latency (ms)	Variability	P(Sig)	P(!Sig)	P(Det|Sig)	P(Det|!Sig)	P(FP)	P(!FP)	P(Det|FP)	P(Det|!FP)
0 - 5	Constant	0.36	0.64	0.37	0.00	0.06	0.94	0.00	0.14
	Noisy	0.18	0.82	0.00	0.10	0.44	0.56	0.20	0.00
	None	0.29	0.71	0.32	0.04	0.17	0.83	0.15	0.11
5 - 25	Constant	0.47	0.53	0.68	0.00	0.04	0.96	0.00	0.33
	Noisy	0.67	0.02	0.55	0.00	0.02	0.98	0.00	0.38
	None	0.57	0.43	0.60	0.00	0.03	0.97	0.00	0.36
25 - 100	Constant	0.80	0.20	0.42	0.00	0.03	0.97	0.00	0.34
	Noisy	0.82	0.18	0.36	0.00	0.02	0.98	0.00	0.30
	None	0.81	0.19	0.38	0.00	0.02	0.98	0.00	0.31
100 - 300	Constant	0.49	0.51	0.26	0.04	0.02	0.98	0.50	0.14
	Noisy	0.59*	0.41*	0.41*	0.05*	0.13*	0.87*	0.18*	0.28*
	None	0.49	0.51	0.26	0.04	0.02	0.98	0.50	0.14
300 +	Constant	0.34	0.66	0.13	0.00	0.05	0.95	0.00	0.05
	Noisy	0.59*	0.41*	0.41*	0.05*	0.13*	0.87*	0.18*	0.28*
	None	0.34	0.66	0.73	0.56	0.05	0.95	0.00	0.65
No Latency	Constant	0.43	0.57	0.32	0.01	0.04	0.96	0.06	0.15
	Noisy	0.59	0.41	0.41	0.05	0.13	0.87	0.18	0.28
	None	0.47	0.53	0.35	0.02	0.06	0.94	0.12	0.18

Table 6.7: Probability values used for the Mode Detector for Bayes' Theorem.
Probability masses marked with an asterisk use the probability samples for the equivalent No Latency category because no samples were available.

One downside of the data fusion approach that became apparent while collecting the ground truth is the manual assignment of severity scores to event groups and calculating the resulting probability masses and values used for each detector is a time consuming, tedious, and error-prone task. Any new detector that is implemented within Netevmon will require its own set of probability masses and values, which may make further detector development an onerous task.

However, the ground truth generated for this work can be reused for this purpose since the events have already been assigned a severity score. However, any new events that were previously undetected by the old detectors will still need to be recorded and assigned a severity score.

Chapter 7

Fusion Method Validation

Having collected the ground truth data and calculated different probability values to use for each detector, a method to compare and validate the different data fusion methods was required. Thus, an event grouping script was created for the purpose of reading event data generated by Netevmon, grouping events together and calculating the final probability values using the data fusion methods. This script was run against a test dataset containing measurements from another set of latency time series where the events have been independently assigned a severity score in the same manner as the original Smokeping ground truth. The resulting event groups and their significance ratings were then compared against the assigned severity scores to evaluate how effective each fusion method is at determining the severity of an event based on the detectors that fired and the properties of the time series.

This chapter describes the implementation and functions of the event grouping script, the test dataset used for evaluating the performance of the data fusion methods and details how the output from the event grouping script is validated against the test dataset.

7.1 Implementation Script

The event grouping script accepts the events generated by the detectors run within Netevmon as input, which includes:

- the timestamp at which the detector fired;
- a unique number to identify the time series that the event came from;
- the name of the detector which triggered the event;

- the type of event (e.g. a mean update, a changepoint event, etc);
- a description of the event;
- the metric of the time series, e.g. latency, byte counts, user counts, etc.

The event grouping script also receives hourly mean updates from Netevmon, which contain the mean of the last 720 measurements for the time series. Additionally, the script receives updates on the variability of the time series from the LatencyTS Detector when a “Noisy to Constant” or “Constant to Noisy” event occurs. The mean and variability updates are used to choose the correct category for the time series based on which categorisation method is being employed, e.g. matching the time series to a category based on mean latency, variability, both, or neither. The categorisation method is chosen when executing the script, allowing each categorisation method to be evaluated easily and independently.

Because each detector will trigger an event at different times for the same change in the time series, the first task of the script is to create an event group containing all the events that are in close proximity, instead of individually processing events from detectors. An event group within this context is simply a grouping of all events that occur within an hour of the first event in the group. Thus, the timestamp of the very first event received by the event grouping script is used as the start of the first event group and any subsequent events that occurred within an hour of the start of the event group are added to the current group. When an event occurs more than an hour away from the current event group’s start time, the probability values for each data fusion method are finalised and the next event group is started with the new event’s timestamp as the start time for that group.

The script has been written with an implementation of each data fusion method which is used to calculate the probability values for each method. The Dempster-Shafer belief values, Bayes’ Theorem and Cumulative Fusion Method posterior probabilities are updated for each group as the events are added to the current group, whereas the final probabilities for the Averaging and Detector Count Heuristic Fusion methods are calculated when an event group is complete. The probability masses and Bayesian likelihoods required for each grouping method and detector are obtained from an external module that contains all of the masses and likelihoods calculated from the Smokeping ground truth, which are included in Appendices A and B. An external module was implemented instead of including the values in the event grouping script since that would allow updating the probability values easily, without having to modify the event grouping script. Additionally, the external module could be used in other projects if required.

Occasionally, a detector may produce multiple events in an event group. In this case, probability values are updated for any duplicate detectors (except for the HMM Detector) because we consider additional events from the same detector within the hour to be a reinforcement of the significance of an event group. The HMM Detector is only accounted for once in every event group because in testing, HMM Detector was observed to be prone to trigger multiple events for insignificant changes in the time series.

While updating the probability values for an event group, the timestamp at which each fusion method recognised that the event group is significant is recorded. Initially, a probability threshold of 0.9 was used to determine whether an event group was significant, i.e. whenever a fusion method reported a probability of significance greater than 0.9. This threshold was chosen with the aim of minimising false positives to ensure that the system would only alert network operators to genuinely significant events, as alerting on insignificant events would reduce the operator's faith in the system. During development of the event grouping script, testing against the Smokeping data suggested that 0.9 was a suitable compromise between the desire for few false positives while still ensuring that most significant events would be identified as such by the system.

Once an event group is finalised, the following details are written as output for each group:

- the timestamp of the first event in the group;
- the number of detectors that fired in that event group;
- the mean latency during that event group;
- the variability of the time series during that event group;
- the probability masses for significance and false positive for the Dempster-Shafer Belief Fusion method;
- the timestamp of the event that caused Dempster-Shafer's significant

probability to exceed the 90% threshold, or 0 if this did not occur;

- the probability values for significance and false positive for Bayes' Theorem Belief Fusion method;
- the timestamp of the event that caused Bayes' Theorem's significant probability to exceed the 90% threshold, or 0 if this did not occur;
- the probability values for significance and false positive for the Averaging Belief Fusion method;
- the timestamp of the event that caused the Averaging method's significant probability to exceed the 90% threshold, or 0 if this did not occur;
- the probability values for significance and false positive for the Cumulative Belief Fusion method;
- the timestamp of the event that caused the Cumulative method's significant probability to exceed the 90% threshold, or 0 if this did not occur;
- the probability value for the event being significant for the Detector Count Heuristic method
- the timestamp at which the Detector Count Heuristic method's significant probability to exceed the 90% threshold, or 0 if this did not occur;
- a list of the detectors in the order in which they fired for that particular event group.

7.2 The Test Dataset

After testing the event grouping script on the training dataset, i.e. the Smokeping latency time series, we evaluated the data fusion methods against a separate dataset. We chose measurements from the AMP-ICMP latency test for this purpose, which consists of latency measurements between a source host and a destination IP address collected using the Active Measurement Project (AMP) [19]. The latency measurements are generated by sending an ICMP Echo Request [33] packet to the target IP address every 30 seconds and waiting for a response from the target. The latency measurement is the delay between the time the echo request was sent and the time the echo response was received by the original sender.

AMP-ICMP was not used as the ground truth data for the purposes of calculating the detector probabilities because it was not deployed widely enough at the time when the ground truth data was collected, which would have limited the size of the sample for the ground truth data. We decided to use AMP-ICMP as the test dataset because Netevmon is part of the Active Measurement Project (AMP), and hence it would be prudent to test the effectiveness of the detectors and data fusion methods on the AMP-ICMP time series. The AMP-ICMP time series also include a wider range of time series using different sources and destinations, whereas the Smokeping time series were sourced from the WAND Network Research Group's internal monitoring and therefore only used a single source and a smaller number of destinations.

The frequency of new measurements in the AMP-ICMP time series is 30 seconds, whereas the frequency of measurements for Smokeping time series is 5 minutes. Because of the shorter delay between measurements for AMP-ICMP time series, it is expected that the delay between the start of an event and a detector identifying the event will be shorter than for Smokeping time series. Additionally, because the individual measurements obtained from the echo requests form the AMP-ICMP time series, there is likely to be much more variation in each time series and this may cause the more sensitive detectors to perform poorly with these time series compared to the Smokeping data. This was not a problem for Smokeping time series because the variability of the individual measurements may have been concealed by a median that was relatively constant.

Ground truth is needed for the test dataset to enable comparing the results of the fusion methods against a set of latency time series events where the significance of the events is already known. The process of collecting and assigning severity scores to the AMP-ICMP test dataset was similar to the method used to acquire the Smokeping ground truth that acted as the training data. The main difference between the AMP-ICMP and Smokeping datasets is that the AMP-ICMP dataset was collected and classified independently by another member of the WAND group. Having another person perform the classification reduced the likelihood of the ground truth for the test dataset being biased based on the observed reliability of each detector during the analysis of the Smokeping dataset. For example, the Mode Detector was shown to be highly reliable for the Smokeping dataset and therefore the person who collected the Smokeping ground truth may subconsciously assume that any event detected by the Mode Detector in the AMP-ICMP dataset is also significant rather than evaluating the event solely on its characteristics.

The AMP-ICMP ground truth dataset includes events from 24 latency time series that were observed during February 2014. Table 7.1 shows a summary of these events. As with the corresponding table for the Smokeping ground truth dataset (Table 6.1), the events are broken down by category and assigned severity scores. The severity scores have the same meaning as before, e.g. a severity score of 0 is a false positive, a score of 1 is an insignificant event, etc. While we endeavoured to ensure that the test dataset included at least 35 significant events for each latency category, it was difficult to find events where the time series was also noisy as few of the high latency series qualified as noisy according to Netevmon. As a result, some Noisy categories are poorly represented in the test dataset, e.g. the 300+ & noisy category only has 10 events in total and only one of those was significant, which limits the number of test cases for those categories.

Latency (ms)	Variability	Contributing Time Series	False Positive Events	Insignificant Events	Undecided Events	Significant Events	Total Events
0 - 5	Noisy	4	0	5	4	13	22
	Constant	4	1	6	2	26	35
5 - 25	Noisy	6	2	18	16	16	52
	Constant	5	2	15	13	26	56
25 - 100	Noisy	5	2	21	10	19	52
	Constant	6	1	60	37	76	174
100 - 300	Noisy	3	4	7	7	7	25
	Constant	7	14	89	32	36	171
300 +	Noisy	1	4	5	0	1	10
	Constant	4	3	37	35	39	114

Table 7.1: The number of false positive, insignificant, undecided and significant events detected in the AMP-ICMP test dataset

Series Source	Series Target	IP Version	False Positive Events	Insignificant Events	Undecided Events	Significant Events	Total Events
inspire	facebook.com	IPv4	0	3	1	1	5
citylink	facebook.com	IPv4	1	7	0	2	10
fx-aknnr	wikipedia.org	IPv6	3	8	5	4	20
lightwire	twitter.com	IPv4	0	10	8	3	21
callplus	youtube.com	IPv6	1	2	1	3	7
massey-pn	netflix.com	IPv6	2	38	12	14	66
waikato	arin.net	IPv6	11	28	12	16	67
inspire	lacnic.net	IPv4	0	6	5	10	21
citylink	lacnic.net	IPv6	0	11	9	14	34
callplus	afrinic.net	IPv6	6	15	9	4	34
ns4a.amp.wand.net.nz	mega.co.nz	IPv4	1	10	12	12	35
Waikato	www.xero.com	IPv4	2	1	12	16	31
catalyst	ns3.dns.net.nz	IPv4	1	20	10	4	35
waikato	www.quickflix.co.nz	IPv4	0	4	4	25	33
lightwire	www.northpower.co.nz	IPv4	0	5	0	10	15
ns4a	www.orcon.co.nz	IPv4	0	3	6	13	22
inspire	csotago	IPv4	2	10	0	5	17
massey-pn	waikato	IPv6	0	1	3	8	12
lightwire	apnic.net	IPv4	2	25	6	9	42
ns4a	8.8.8.8	IPv4	0	10	9	21	40
fx-aknnr	www.vocus.com.au	IPv6	0	8	4	1	13
waikato	www.vocus.com.au	IPv6	0	8	6	3	17
lambda	h.root-servers.net	IPv6	1	22	6	7	36
catalyst	cloudflare.com	IPv4	0	17	4	5	26

Table 7.2: The number of false positive, insignificant, undecided and significant events events detected in each of the individual AMP-ICMP time series in the test dataset

Table 7.2 contains a list of all the AMP-ICMP time series present in the test dataset and the events detected for each time series, categorised by manually assigned significance. For AMP-ICMP, a “Series Source” is a location where an AMP monitor has been deployed, e.g. an ISP, a university, a data centre. A “Series Target” can be an AMP monitor, a website or a DNS server that the “Series Source” tests to. The latency and variability have been excluded from this table because the mean and variability often differed greatly during the examined period, which meant that a series would not have a regular variability or mean latency. Instead, the mean and variability was recorded for each individual event, rather than having a single category value for the entire time series.

Table 7.3 shows the mean delay between the start of an event and the timestamp when each detector identified the event in the AMP ICMP dataset. The delays are much smaller for the AMP-ICMP time series than for the Smokeping time series because of the higher measurement frequency of 30 seconds, whereas the frequency for the Smokeping latency time series was much lower at 5 minutes. While the average delay in minutes might be lower for the AMP-ICMP time series, the number of measurements required before the event is detected is slightly higher than the Smokeping time series because the noisiness of the time series makes it difficult to fulfil some of the conditions required for triggering an event. For example, the Mode detector would take longer to identify a distinct mode when having to consider the full range of observed measurements rather than the median and hence, take more measurements to detect anomalous behaviour in the time series.

	Severity Score					
Detector	1	2	3	4	5	All Events
Plateau	6.1	8.3	8.6	10.3	7.0	8.2
Changepoint	5.3	6.3	8.5	8.1	5.6	6.2
TEntropy-StdDev	7.0	6.2	6.8	9.2	7.1	6.9
TEntropy-MeanDiff	7.1	7.1	8.0	10.8	7.7	7.8
Mode	N/A	7.9	9.4	8.4	8.9	8.8
HMM	3.9	4.9	5.1	6.7	4.3	4.7
All Detectors	5.8	6.5	7.8	9.0	6.7	6.8

Table 7.3: Summary of the average delay in minutes before a detector fires for events found in the AMP-ICMP test dataset

7.3 Validation Method

Now that we have a pre-classified test dataset where the severity scores have been manually assigned, it is now possible to compare the detector probabilities produced by the event grouping script against the test dataset. An event matching script was created for this purpose. The script initially loads the manually classified event groups from the test dataset, including the timestamp when each event started and the timestamps at which each detector reported the event. The results from the event grouping script are then loaded, including the timestamp of the first event in the group, the time series mean, variability of the time series at that time, a set of probabilities calculated using each fusion method (e.g. Bayes Sig and Bayes FP) and a timestamp for each fusion method for when the significance threshold of 90% was reached. The timestamp of the first event from the event grouping script is compared with the timestamps at which the individual detectors fired from the ground truth data. When a match is found, the event group from the test dataset is removed from the available ground truth events and the output string is constructed for that particular event group, which includes:

- a hyphenated string of the source, target and IP version of the AMP-ICMP time series for a particular event, e.g. waikato-arin-ipv6;
- the timestamp when the manually classified event group started;
- the severity score manually assigned to that particular event group;
- the number of detectors that fired for the event group, according to the ground truth;
- the timestamp of the first event in the group observed by the event grouping script;
- the number of detectors that fired for that event group, according to the event grouping script;
- the mean latency of the time series during that particular event group;
- the variability of the time series during that particular event group;
- the probabilities of the event being significant, reported by each of the fusion methods;
- the probabilities of the event being a false positive, reported by each of

the fusion methods;

- the timestamp of the event that caused each individual fusion method's significant probability to exceed the 90% significance threshold, or 0 if the probability did not exceed the significance threshold for that fusion method;
- a list of the detectors in the order in which they fired for that particular event, primarily used for debugging.

Ideally, this process would continue until all of the event groups in the test dataset have been matched to a corresponding set of probabilities from the event grouping script. However, because Netevmon and the detectors are under constant development, the events produced when the AMP-ICMP ground truth was collected differed slightly from the events observed when the validation method was run. Some detectors, specifically the HMM Detector and the TEntropy Detectors, also vary their results slightly depending on the measurements they received in their training phase. This resulted in the manually classified events not having exactly the same timestamps as the events produced with the event grouping script.

As a solution to this problem, we added another rule to identify a match when the timestamp of the first event from the ground truth is within 5 minutes of the timestamp of the first event from the event grouping script. This ensures that event groups will be matched to a set of probabilities even if the timestamp at which the detectors fired differ slightly.

The final stage of the validation is to determine if each fusion method has correctly identified the significance of the event group as described in the ground truth for the test dataset. If the manually assigned severity score for an event group is less than 3, i.e. the event is not significant, a correct classification would require that the probability of significance never exceeds the threshold of 90%. Otherwise, the opposite must occur, i.e. the probability of significance must exceed 90% for that event group at some point during the lifetime of the group.

The best fusion method will be the method that not only correctly classifies the highest number of significant events, but also classifies the least number of insignificant events as significant.

Chapter 8

Results

This chapter presents the results of running each data fusion method against the AMP-ICMP test dataset and comparing those results against the known ground truth for the test dataset. The accuracy of each fusion method will be shown and the best data fusion method for identifying significant anomalies in a latency time series will be determined.

As mentioned in Chapter 6, four different categorisation methods were used when evaluating the data fusion methods. A categorisation method is used to determine which set of probabilities to use for an event group. For example, an event group with a mean latency of 20 and a variability of “Noisy” will use probabilities from the “5 - 25 and Noisy” category if the “Latency and Variability” categorisation method is used. The same event group would use probabilities from the “5 - 25 and None” category if the “Latency” categorisation method was used instead. As well as evaluating the accuracy of the data fusion methods, we also aimed to determine the best categorisation method for use with each data fusion method. In particular, does the latency or variability of the time series matter when deriving mass functions or prior probabilities for the anomaly detection methods?

Figures 8.1 to 8.8 show the accuracy of classifying events using four categorisation methods and the five data fusion methods used in this project. The events were grouped by their severity scores, resulting in six sets of bars (labelled 0 to 5) and these represent the accuracy of each data fusion method at correctly classifying events matching each of the different severity scores. The data fusion methods represented by the bars from left to right are: Detector Count Heuristic, Dempster-Shafer, Bayes’ Theorem, Averaging Belief Fusion and Cumulative Belief Fusion.

The ideal result for any data fusion method is 100% accurately classified events, which means that all events with a severity score from 0 to 2 are correctly classified classified as insignificant, while all events with a severity score from 3 to 5 are correctly identified as significant by the fusion method. If the percentage of accurately classified events for any given data fusion method and severity score is below 100%, this means that the data fusion method has produced false positives (for events with a severity score of 0, 1 or 2) or false negatives (for events with a severity score of 3, 4 or 5).

Figure 8.1 displays the results of using the “Latency and Variability” categorisation method with a significance threshold of 0.9. While the rate of correct classifications for significant events is much lower than the goal of 100%, all of the data fusion methods have a relatively low false positive rate. Averaging Belief Fusion was the best fusion method for correctly classifying insignificant events, but was very poor at recognising significant events. By contrast, Dempster-Shafer was the best method for accurate classification of significant events overall but also produced the highest number of false positives, marginally ahead of Bayes’ Theorem. However, the accuracy rate for Dempster-Shafer is poor overall: only 60% of events with a severity score of 5 were classified as significant and the accuracy was even lower for severity 3 and 4.

Figure 8.2, 8.3 and 8.4 display the results of using the “Latency”, “Variability” and “No Categorisation” methods respectively. The “Latency” categorisation method shown in Figure 8.2, has the least number of false positives but also the greatest number of false negatives. While most of the data fusion methods for the “No Categorisation” method shown in Figure 8.4 performed similarly to the “Latency and Variability” classification method, Dempster-Shafer improved quite noticeably in terms of reducing false negatives (80% of events with a severity score of 5 correctly classified), while the other data fusion methods observed only minor changes.

The best results overall were achieved when using the “Variability” classification method. Dempster-Shafer, Bayes’ Theorem and Cumulative Belief Fusion all achieved a classification accuracy of greater than 70% for events with a severity score of 5, with Dempster-Shafer having the highest accuracy at 85%. False negatives were also decreased for events with a severity score of 3 or 4.

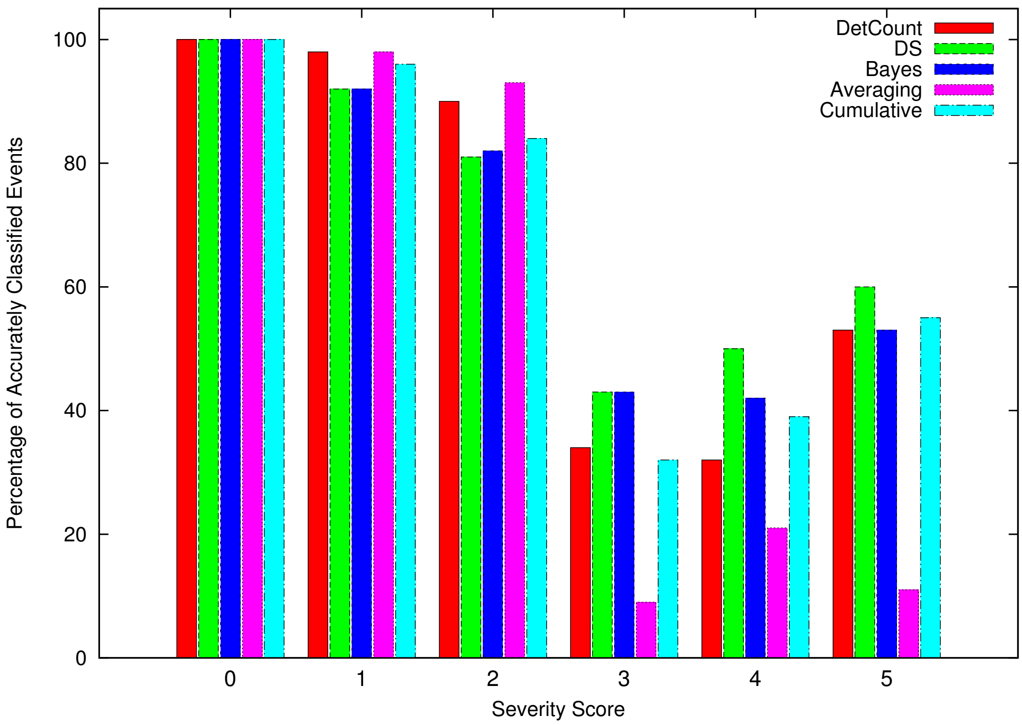

Figure 8.1: Accuracy when using detector probabilities for the Latency and Variability categorisation method with a significance threshold of 0.9

However, these gains are partly counteracted by an increased false positive rate, especially for Dempster-Shafer. It appears that the variability of a time series has the biggest impact on the performance of the detectors, whereas accounting for latency actually decreased the accuracy of Dempster-Shafer and Bayes' Theorem.

From these results, we suggest that Dempster-Shafer is the best performing data fusion method. While the other fusion methods have a lower false positive rate, Dempster-Shafer achieves a much lower false negative rate than the other methods, especially when using the "Variability" categorisation method. The main weakness of Dempster-Shafer is a high false positive rate among events with a severity score of 2. However, these events are typically events that are neither clearly insignificant or significant and therefore may be of some interest to the network operator. As a result, a higher false positive rate among events with a severity score of 2 is not a critical flaw for Dempster-Shafer, although it warrants further investigation to see if there is a way for the false positive rate to be reduced.

While Bayes' Theorem and the Cumulative Belief Fusion methods perform similarly, or even better, to Dempster-Shafer when classifying insignificant events correctly, they are surpassed at identifying significant events by Dempster-Shafer most of the time. The Detector Count Heuristic method had a low rate of false positives, but performed poorly when classifying significant events. This suggests that simply relying on the number of detectors that fired does not produce accurate results.

The worst data fusion method is the Averaging Belief Fusion, which works well for identifying insignificant events as insignificant, but performs very poorly at accurately determining when events are significant. As stated earlier, this is not an unexpected result as the Averaging Belief Fusion method is better suited for dependent evidence, i.e. multiple sources of evidence that depend on each other for their results, whereas the detectors deployed by Netevmon are independent.

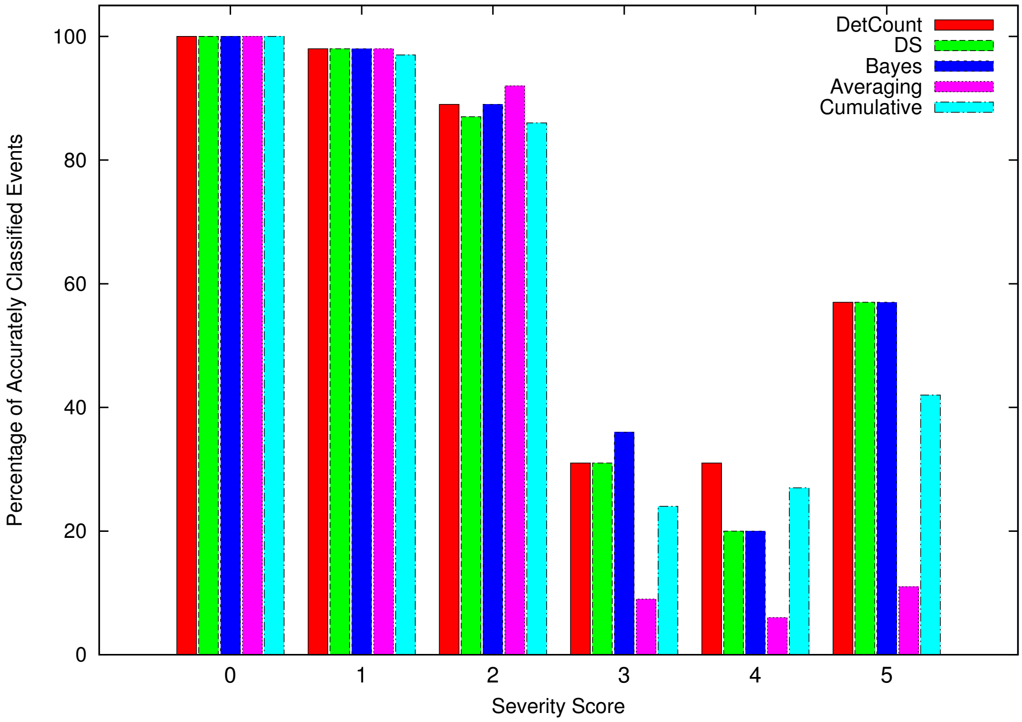

Figure 8.2: Accuracy when using detector probabilities for the Latency categorisation method with a significance threshold of 0.9

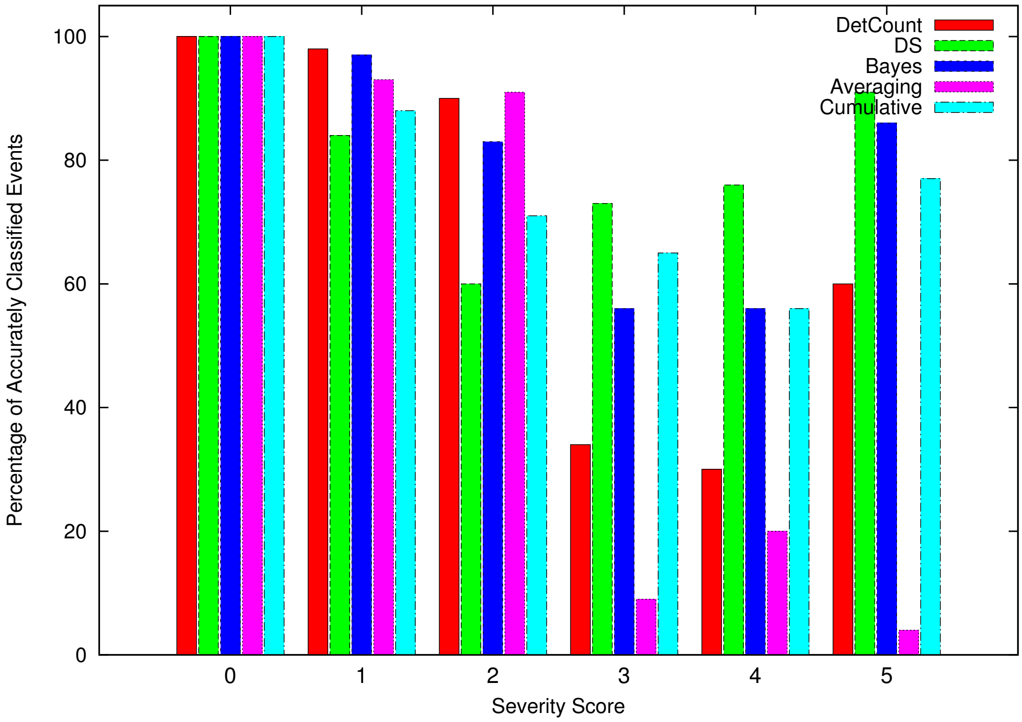

Figure 8.3: Accuracy when using detector probabilities for the Variability categorisation method with a significance threshold of 0.9

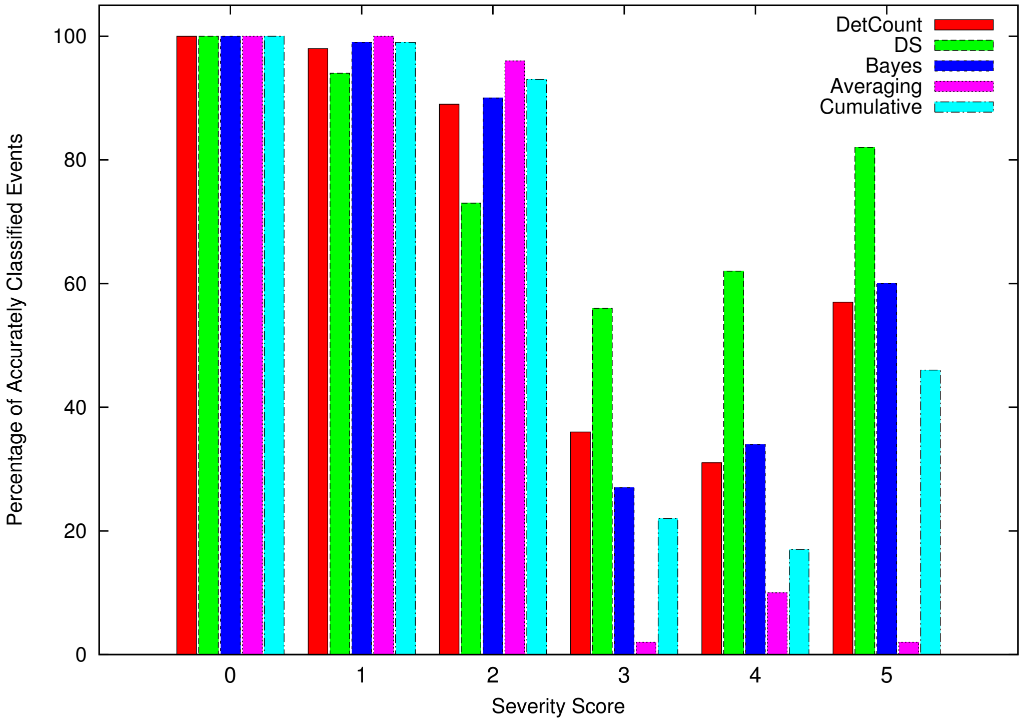

Figure 8.4: Accuracy when using detector probabilities for the No Categorisation method with a significance threshold of 0.9

One possible reason why the "Latency and Variability" categorisation method performed so poorly was that splitting the events in the time series used as training data by latency and variability may have resulted in sample sizes that were too small to generate a representative belief or probability. A larger sample size might produce better results with the "Latency and Variability" categorisation method, but collecting a larger sample for each category would be prohibitive because of the tedious ground truth processing and the difficulty in finding appropriate test targets for some categories.

Because all of the methods produced a high false negative rate that is unsuitable for practical use, we considered the possibility that the significance threshold might have been too high, making it too difficult for the fusion methods to recognise an event as significant. To test this, we lowered the significance threshold from 0.9 to 0.85 and repeated the validation experiment.

Figures 8.5 to 8.8 display the results of using the four categorisation methods with a significance threshold of 0.85. While the "Variability" categorisation method was clearly the best performer at correctly classifying events when using a significance threshold of 0.9, this distinction is harder to make for the results when using a lower significance threshold. The "Variability" categorisation method was again the worst performer in terms of the number of false positives, while being the best categorisation method at identifying significant events across all data fusion methods. There was a small improvement in accuracy when classifying significant events across all of the fusion methods. For example, when using the "Variability" categorisation method, the percentage of correctly classified events with a severity score of 3 and a significance threshold of 0.85 are 76%, 68%, 20%, 73%, and 36% as opposed to 73%, 56%, 9%, 65% and 34% for Dempster-Shafer, Bayes' Theorem, Averaging Belief Fusion, Cumulative Belief Fusion and Detector Count Heuristics respectively. However, we also observed a significant increase in the number of false positives for severity score 2 for all the fusion methods. For example, the percentage of false positives produced by Dempster-Shafer was very high at 49% when using a significance threshold of 0.85, whereas a threshold of 0.9 produced a lower percentage of false positives (40%).

Overall, lowering the significance threshold has had the expected effect: the number of false negatives has decreased, but the number of false positives has increased. The increase in false positives is much larger than the improvement in the number of false negatives, suggesting that 0.9 is a better significance threshold.

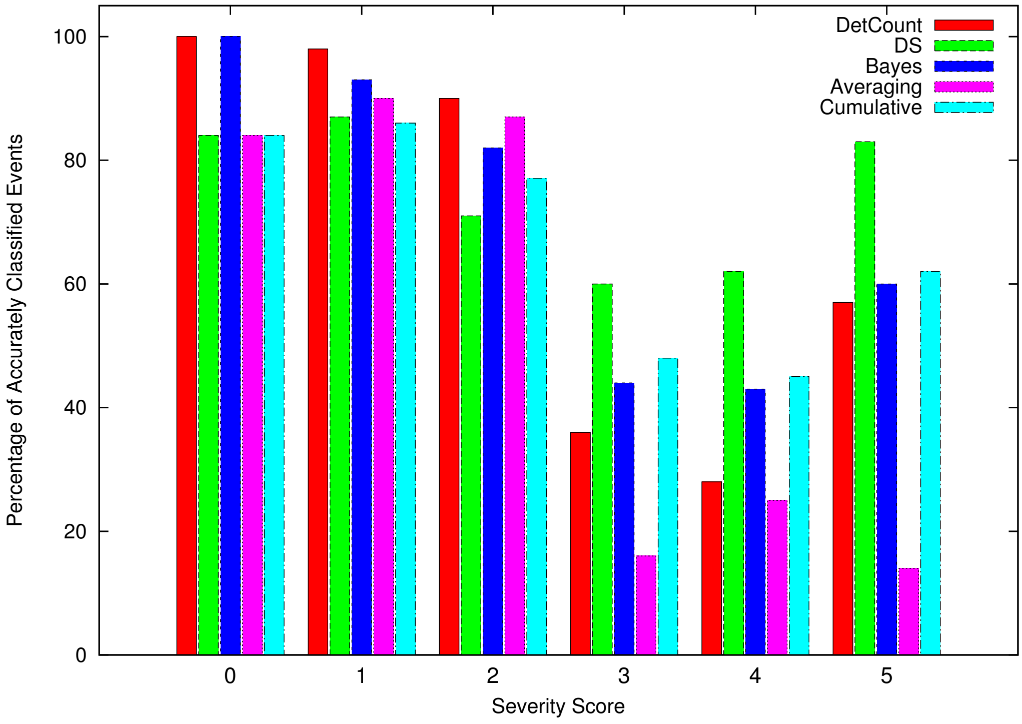

Figure 8.5: Accuracy when using detector probabilities for the Latency and Variability categorisation method with a significance threshold of 0.85

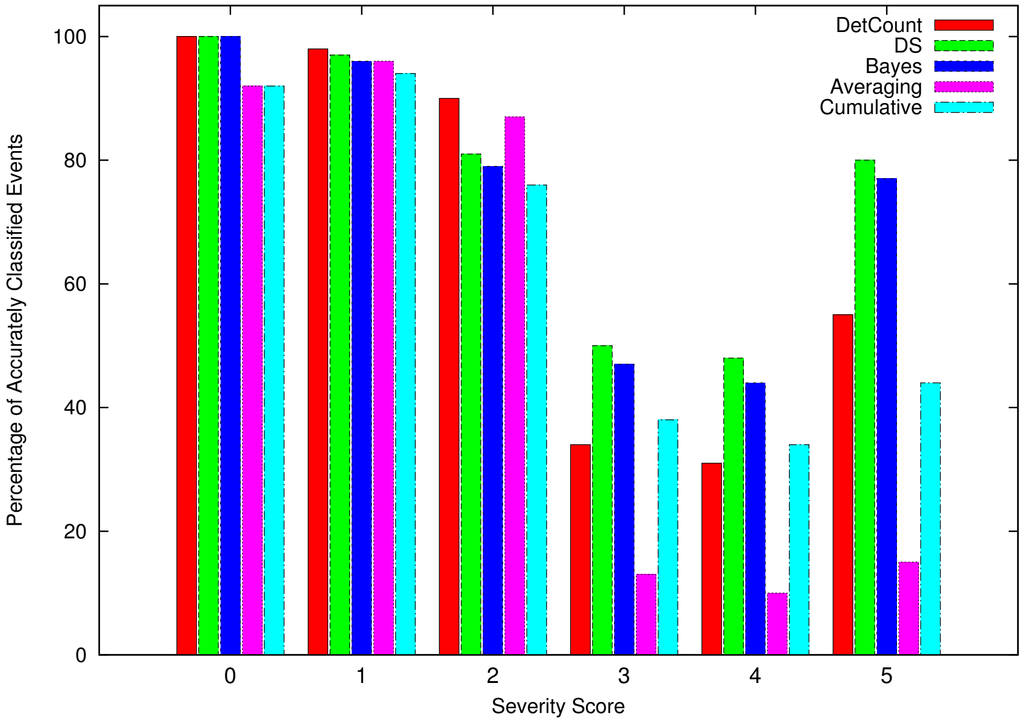

Figure 8.6: Accuracy when using detector probabilities for the Latency categorisation method with a significance threshold of 0.85

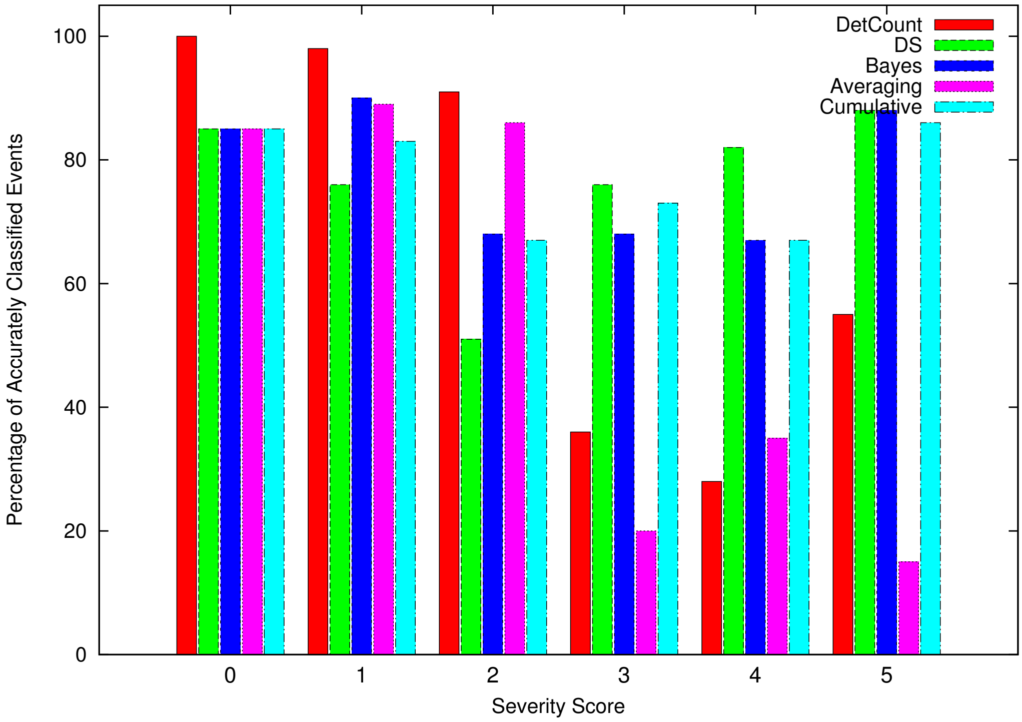

Figure 8.7: Accuracy when using detector probabilities for the Variability categorisation method with a significance threshold of 0.85

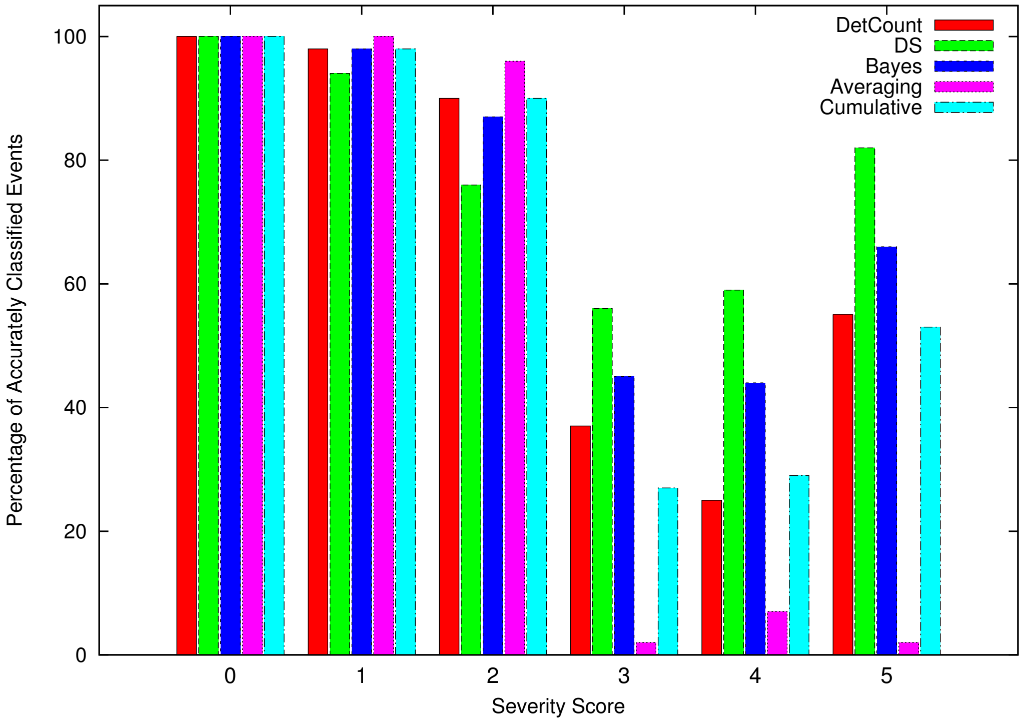

Figure 8.8: Accuracy when using detector probabilities for the No Categorisation method with a significance threshold of 0.85

In Figure 8.7, the false negative rate for Dempster-Shafer decreased by 3% and 6% for events with a severity score of 3 and 4 respectively. For severity score 5, the false negative rate actually increased by 3% with the lower threshold. By contrast, the false positive rate increased by 15%, 8% and 9% for events with a severity score of 0, 1, and 2 respectively. Similar results are seen for Bayes' Theorem and Cumulative Belief Fusion, where any improvements in the false negative rate are counteracted by an equal or larger increase in the false positive rate. Lowering the threshold is more helpful when using the other categorisation methods, e.g. the "Latency" categorisation method, but the resulting accuracy is still inferior to the accuracy of the "Variability" categorisation method.

8.1 Possible Improvements

None of the fusion methods achieved a satisfactory level of accuracy, especially when considering the rate of false negatives. We have shown that decreasing the significance threshold from 0.9 to 0.85 resulted in a larger rate of false positives, while only providing a small decrease in the rate of false negatives, so this is not a viable solution. However, there are several options available that may improve the classification accuracy which could form the basis for further research.

Collecting additional ground truth and increasing the sample size of the test dataset would ensure that each of the categories for the different categorisation methods has a representative sample to eliminate bias towards a particular time series, which could occur when a category contains events from a single time series only. Some categories had very little training data available and in these cases, rather than the probability representing a detector's reliability for the category, it represents the reliability for the time series. However, latency time series for the under-represented categories can be hard to find, even with the variety of targets tested by AMP.

Some significant events in the AMP-ICMP dataset were only detected by one or two detectors, which means that a lack of evidence often prevents probabilities from crossing the significance threshold. Implementing more detectors within Netevmon may help, but this requires a lot of time and effort to research and implement new detectors and then to generate the probabilities for the fusion methods. Fortunately, the ground truth and manually assigned severity scores from the Smokeping and AMP-ICMP dataset can be used for this purpose, reducing the workload somewhat. An additional improvement could be to incorporate evidence from other time series modules, rather than just latency alone. For example, if a change in the Traceroute path is observed at the same time as a latency event, that could act as further evidence in favour of the event being significant.

Another problem was that the probabilities used by the fusion methods were based on Smokeping time series, but our test data was from AMP. We had assumed that because they were both latency time series, the detector performance would be similar and thus, reusing the Smokeping probabilities for AMP latency time series would be suitable. However, it appears that the AMP time series would benefit from having their own set of probabilities for each detector, especially given the higher variability of AMP-ICMP time series and the likely impact on sensitive detection methods. One avenue of future work would be to replace the probabilities and mass functions with values derived from AMP-ICMP events and evaluate whether there is any improvement in the accuracy of classifying events. However, this would require collecting a new test dataset (using either new AMP-ICMP time series or different time periods from the same time series) because the previous test dataset is now being used as training data.

Chapter 9

Conclusion

This report has presented an evaluation of five data fusion methods (Dempster-Shafer, Bayes' Theorem, Averaging Belief Fusion, Cumulative Belief Fusion, and Detector Count Heuristic) for the purpose of identifying significant events in a latency time series. The aim of this work was to identify a method that can reliably indicate to a network operator whether a detected anomaly requires immediate attention.

Six anomaly detection methods implemented within Netevmon, an anomaly detection framework for time series data, were used as sources of evidence for the data fusion techniques. Four of the detection methods were researched and implemented as part of this project because data fusion requires multiple sources of evidence to be effective and only two detectors were implemented within Netevmon at the start of this work. To combine the evidence, the data fusion methods required a set of probabilities describing the likelihood that a detected event is significant for each of the anomaly detection methods. This was achieved using ground truth data collected for 535 events from 25 Smokeping time series. Each event was manually assigned a severity score of 0 to 5, reflecting the perceived significance of the event and the probabilities were derived from this ground truth. The latency and variability of the time series were used to divide the events into categories based on four different categorisation methods for the purpose of evaluating the impact of the latency and variability of a time series on the reliability of the detectors.

Another dataset of 24 AMP-ICMP latency time series was collected for evaluating the accuracy of the data fusion methods. Ground truth was also obtained for the test dataset by manually processing the AMP-ICMP time series using the same process that was employed with the Smokeping training dataset. Events from the test dataset were passed through the data fusion methods and the resulting belief score, i.e. a fusion method's belief that an event was significant, was compared against the manually assigned severity scores for the test dataset. A significance threshold of 0.9 was used to determine if an event was significant or not. An insignificant event (severity scores 0, 1, or 2) had been correctly classified when the probability of significance was below the significance threshold. Significant events (severity scores 3, 4 or 5) had been correctly classified if the probability of significance reached or exceeded 0.9.

The results of the evaluation experiments showed that the accuracy of all of the data fusion methods was unsuitable for practical deployment. The results also showed that the variability of a time series has a noticeable impact on the reliability of the detectors, whereas calculating separate probabilities for different latency ranges made negligible improvements to the accuracy of the fusion methods. After analysing the results, we concluded that Dempster-Shafer was the most accurate data fusion method as it achieved the least number of false negatives, despite having a slightly higher false positive rate than Bayes' Theorem and Cumulative Belief Fusion. From this, we suggest that future research in this area should be centered on improving the accuracy of the Dempster-Shafer method to reach a level where it would be useful for network operators.

We have identified a number of possible approaches towards decreasing the rate of incorrect significance classifications. The first possible improvement would be to collect additional ground truth data to ensure that the derived probabilities are based on a representative sample, instead of being biased towards a particular time series. Implementing a wider variety of detectors within Netevmon would provide additional evidence for events being significant because many of the false negatives in the AMP-ICMP test dataset were only reported by one or two detectors. Using evidence from the other Time Series modules would also be a useful addition because some events will also be apparent in other network metrics, e.g. routing changes can affect both the latency and the Traceroute path. Another approach would be to derive the probabilities for the fusion methods from the AMP-ICMP dataset because we believe that the high variability of the AMP-ICMP time series may have had a detrimental effect on the performance of some of the more sensitive detectors. This approach would require the collection of a new independent dataset because the original AMP-ICMP test dataset would be used as training data.

References

[1] A Simple Network Management Protocol (SNMP). http://www.ietf.org/rfc/rfc1157.txt. Accessed: 2014.05.23.

[2] Intrusion Detection Message Exchange Format Data Model and Extensible Markup Language (XML) Document Type Definition. http://www.ietf.org/proceedings/50/I-D/idwg-idmef-xml-03.txt. Accessed: 2014.05.23.

[3] The Libpcap Project. http://sourceforge.net/projects/libpcap/. Accessed: 2014.05.23.

[4] Ryan Prescott Adams and David JC MacKay. Bayesian Online Changepoint Detection. *arXiv preprint arXiv:0710.3742*, 2007.

[5] G. Androulidakis, V. Chatzigiannakis, and S. Papavassiliou. Network Anomaly Detection and Classification via Opportunistic Sampling. *Network, IEEE*, 23(1):6–12, January 2009.

[6] Thomas Bayes, Derek W Bunn, Howard Raiffa, Robert Schlaifer, and Detlof Von Winterfeldt. An Essay toward Solving a Problem in the Doctrine of Chances. *Philosophical Transactions of the Royal Society of London*, 53, 1984.

[7] Peter J Brockwell and Richard A Davis. *Introduction to Time Series and Forecasting*, volume 1. Taylor & Francis, 2002.

[8] SC Byun, DB Choi, BH Ahn, and Hanseok Ko. Traffic Incident Detection using Evidential Reasoning Based Data Fusion. In *Proceedings of 6th World Congress on Intelligent Transport Systems (ITS), held Toronto, Canada, November 8-12, 1999*, 1999.

[9] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly Detection: A Survey. *ACM Comput. Surv.*, 41(3):15:1–15:58, July 2009.

[10] V. Chatzigiannakis, G. Androulidakis, K. Pelechrinis, S. Papavassiliou,

and V. Maglaris. Data Fusion Algorithms for Network Anomaly Detection: Classification and Evaluation. In *Proceedings of the Third International Conference on Networking and Services*, ICNS '07, pages 50–, Washington, DC, USA, 2007. IEEE Computer Society.

[11] Vasilis Chatzigiannakis, Symeon Papavassiliou, Georgios Androulidakis, and B Maglaris. On the Realization of a Generalized Data Fusion and Network Anomaly Detection Framework. *Fifth International Symposium on Communication Systems, Networks and Digital Signal Processing (CSNDSP'06), Patra, Greece*, 2006.

[12] Siddhartha Chib. Estimation and Comparison of Multiple Change-point Models. *Journal of econometrics*, 86(2):221–241, 1998.

[13] Can Demirkesen and Hocine Cherifi. Fusing Image Representations for Classification Using Support Vector Machines. *CoRR*, abs/1207.3607, 2012.

[14] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38, 1977.

[15] Yong Deng, Xiaoyan Su, Dong Wang, and Qi Li. Target Recognition Based on Fuzzy Dempster Data Fusion Method. *Defence Science Journal*, 60(5):525–530, 2010.

[16] Werner Ebeling, Ralf Steuer, and MR Titchener. Partition-based Entropies of Deterministic and Stochastic Maps. *Stochastics and Dynamics*, 1(01):45–61, 2001.

[17] Nagios Enterprises. Nagios. http://www.nagios.org/. Accessed on: 2014.05.05.

[18] Terrence L. Fine. Review: Glenn shafer, a mathematical theory of evidence. *Bulletin of the American Mathematical Society*, 83(4):667–672, 07 1977.

[19] WAND Network Group. Active Measurement Project. http://research.wand.net.nz/software/amp.php. Accessed: 2014.06.06.

[20] WAND Network Research Group. WAND Network Research Group. http://wand.net.nz/. Accessed: 2014.05.24.

[21] Ian Jolliffe. *Principal Component Analysis*. Wiley Online Library, 2005.

[22] Audun Jøsang, Javier Diaz, and Maria Rifqi. Cumulative and Averaging Fusion of Beliefs. *Inf. Fusion*, 11(2):192–200, April 2010.

[23] Shrijit S. Joshi and Vir V. Phoha. Investigating Hidden Markov Models Capabilities in Anomaly Detection. In *Proceedings of the 43rd Annual Southeast Regional Conference - Volume 1*, ACM-SE 43, pages 98–103, New York, NY, USA, 2005. ACM.

[24] Christopher Krügel, Thomas Toth, and Engin Kirda. Service Specific Anomaly Detection for Network Intrusion Detection. In *Proceedings of the 2002 ACM Symposium on Applied Computing*, SAC '02, pages 201–208, New York, NY, USA, 2002. ACM.

[25] Tze Leung Lai. Sequential Changepoint Detection in Quality Control and Dynamical Systems. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 613–658, 1995.

[26] Jiaming Li, Suhuai Luo, and Jesse S. Jin. Sensor Data Fusion for Accurate Cloud Presence Prediction Using Dempster-Shafer Evidence Theory. *Sensors*, 10(10):9384–9396, 2010.

[27] Wei Lu and Ali A. Ghorbani. Network Anomaly Detection Based on Wavelet Analysis. *EURASIP J. Adv. Signal Process*, 2009:4:1–4:16, January 2009.

[28] Andreas Löf. *Improving the Evaluation of Network Anomaly Detection Using a Data Fusion Approach*. PhD thesis, University of Waikato, 2013.

[29] A.J. McGregor and Braun H-W. Automated Event Detection for Active Measurement Systems, 2001.

[30] JJ O Ruanaidh, WJ Fitzgerald, and KJ Pope. Recursive Bayesian Location of a Discontinuity in Time Series. In *Acoustics, Speech, and Signal Processing, 1994. ICASSP-94., 1994 IEEE International Conference on*, volume 4, pages IV–513. IEEE, 1994.

[31] Tobi Oetiker. About Smokeping. http://oss.oetiker.ch/smokeping/index.en.html. Accessed: 2014.05.05.

[32] Vern Paxson. Bro: a System for Detecting Network Intruders in Real-Time. *Computer Networks*, 31(23-24):2435–2463, 1999.

[33] J. Postel. Internet Control Message Protocol. RFC 792 (INTERNET STANDARD), September 1981. Updated by RFCs 950, 4884, 6633, 6918.

[34] Michael J Prerau and Eleazar Eskin. Unsupervised Anomaly Detection using an Optimized K-nearest Neighbors Algorithm. *Undergraduate Thesis, Columbia University: December*, 2000.

[35] Lawrence R Rabiner. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[36] Niko Rebenich. Fast Low Memory T-Transform. http://www.t-codes.org/. Accessed: 2014.05.05.

[37] Martin Roesch. Snort - Lightweight Intrusion Detection for Networks. In *Proceedings of the 13th USENIX Conference on System Administration*, LISA '99, pages 229–238, Berkeley, CA, USA, 1999. USENIX Association.

[38] R. Sekar, A. Gupta, J. Frullo, T. Shanbhag, A. Tiwari, H. Yang, and S. Zhou. Specification-based Anomaly Detection: A New Approach for Detecting Network Intrusions. In *Proceedings of the 9th ACM Conference on Computer and Communications Security*, CCS '02, pages 265–274, New York, NY, USA, 2002. ACM.

[39] Glenn Shafer. *A Mathematical Theory of Evidence*, volume 1. Princeton university press Princeton, 1976.

[40] Taeshik Shon, Yongdae Kim, Cheolwon Lee, and Jongsub Moon. A Machine Learning Framework for Network Anomaly Detection using SVM and GA. In *Information Assurance Workshop, 2005. IAW '05. Proceedings from the Sixth Annual IEEE SMC*, pages 176–183, June 2005.

[41] Taeshik Shon and Jongsub Moon. A Hybrid Machine Learning Approach to Network Anomaly Detection. *Information Sciences*, 177(18):3799 – 3821, 2007.

[42] Christos Siaterlis and Basil Maglaris. Towards Multisensor Data Fusion for DoS Detection. In *Proceedings of the 2004 ACM Symposium on Applied Computing*, SAC '04, pages 439–446, New York, NY, USA, 2004. ACM.

[43] A. F. M. Smith. A Bayesian Approach to Inference about a Change-point in a Sequence of Random Variables. *Biometrika*, 62(2):407–416, 1975.

[44] U. Speidel, R. Eimann, and N. Brownlee. Detecting Network Events via T-entropy. In *Information, Communications Signal Processing, 2007 6th International Conference on*, pages 1–5, Dec 2007.

[45] DA Stephens. Bayesian Retrospective Multiple-Changepoint Identification. *Applied Statistics*, pages 159–178, 1994.

[46] M. Thottan and Chuanyi Ji. Anomaly Detection in IP Networks. *Signal Processing, IEEE Transactions on*, 51(8):2191–2204, Aug 2003.

[47] K Tomsovic and B Baer. Fuzzy Information Approaches to Equipment Condition Monitoring and Diagnosis. *Electric Power Applications of Fuzzy Systems, IEEE Press*, pages 59–84, 1998.

[48] DW Trigg and AG Leach. Exponential Smoothing with an Adaptive Response Rate. *Journal of the Operational Research Society*, 18(1):53–59, 1967.

[49] A. Ziviani, A. T A Gomes, M.L. Monsores, and P.S.S. Rodrigues. Network Anomaly Detection using Nonextensive Entropy. *Communications Letters, IEEE*, 11(12):1034–1036, December 2007.

Appendix A

Probability Masses for Dempster-Shafer Belief Fusion

Probability masses marked with an asterisk use the probability samples for the equivalent No Latency category because no samples were available.

Latency (ms)	Variability	m(DetSig)	m(DetFP)	m(DetAny)
0 - 5	Constant	0.64	0.00	0.36
	Noisy	0.00	0.00	1.00
	None	0.59	0.00	0.41
5 - 25	Constant	0.89	0.00	0.11
	Noisy	0.00	0.00	1.00
	None	0.95	0.00	0.05
25 - 100	Constant	0.93	0.00	0.07
	Noisy	0.91	0.00	0.09
	None	0.92	0.00	0.08
100 - 300	Constant	0.69	0.00	0.31
	Noisy	0.91*	0.00*	0.09*
	None	0.69	0.00	0.31
300 +	Constant	0.37	0.00	0.63
	Noisy	0.91*	0.00*	0.09*
	None	0.37	0.00	0.63
No Latency	Constant	0.61	0.00	0.39
	Noisy	0.91	0.00	0.09
	None	0.67	0.00	0.33

Table A.1: Probability masses used for the Plateau Detector for Dempster-Shafer, Averaging, and Cumulative Belief Fusions

Latency (ms)	Variability	m(DetSig)	m(DetFP)	m(DetAny)
0 - 5	Constant	0.43	0.00	0.57
	Noisy	0.00	0.00	0.00
	None	0.43	0.00	0.57
5 - 25	Constant	0.36	0.05	0.59
	Noisy	0.89	0.00	0.11
	None	0.52	0.03	0.45
25 - 100	Constant	0.89	0.11	0.00
	Noisy	0.86	0.00	0.14
	None	0.88	0.06	0.06
100 - 300	Constant	0.78	0.06	0.17
	Noisy	0.88*	0.00*	0.13*
	None	0.78	0.06	0.17
300 +	Constant	0.39	0.25	0.36
	Noisy	0.88*	0.00*	0.13*
	None	0.39	0.25	0.36
No Latency	Constant	0.52	0.11	0.37
	Noisy	0.88	0.00	0.13
	None	0.57	0.09	0.34

Table A.2: Probability masses used for the Changepoint Detector for Dempster-Shafer, Averaging, and Cumulative Belief Fusions

Latency (ms)	Variability	m(DetSig)	m(DetFP)	m(DetAny)
0 - 5	Constant	0.81	0.00	0.19
	Noisy	0.17	0.33	0.50
	None	0.58	0.12	0.30
5 - 25	Constant	0.82	0.00	0.18
	Noisy	0.71	0.00	0.29
	None	0.76	0.00	0.24
25 - 100	Constant	0.82	0.00	0.18
	Noisy	0.84	0.00	0.16
	None	0.83	0.00	0.17
100 - 300	Constant	0.47	0.02	0.51
	Noisy	0.88*	0.00*	0.12*
	None	0.47	0.02	0.51
300 +	Constant	0.40	0.00	0.60
	Noisy	0.88*	0.00*	0.12*
	None	0.40	0.00	0.60
No Latency	Constant	0.52	0.11	0.37
	Noisy	0.88	0.00	0.12
	None	0.57	0.09	0.34

Table A.3: Probability masses used for the Tentropy-StdDev Detector for Dempster-Shafer, Averaging, and Cumulative Belief Fusions

Latency (ms)	Variability	m(DetSig)	m(DetFP)	m(DetAny)
0 - 5	Constant	0.41	0.08	0.52
	Noisy	0.00	0.62	0.38
	None	0.34	0.17	0.49
5 - 25	Constant	0.70	0.00	0.30
	Noisy	0.66	0.03	0.31
	None	0.67	0.01	0.31
25 - 100	Constant	0.89	0.00	0.11
	Noisy	0.92	0.00	0.08
	None	0.91	0.00	0.09
100 - 300	Constant	0.84	0.00	0.16
	Noisy	0.68*	0.11*	0.22*
	None	0.84	0.00	0.16
300 +	Constant	0.95	0.00	0.05
	Noisy	0.68*	0.11*	0.22*
	None	0.95	0.00	0.05
No Latency	Constant	0.65	0.03	0.31
	Noisy	0.68	0.11	0.22
	None	0.66	0.06	0.28

Table A.4: Probability masses used for the TEntropy-MeanDiff Detector for Dempster-Shafer, Averaging, and Cumulative Belief Fusions

Latency (ms)	Variability	m(DetSig)	m(DetFP)	m(DetAny)
0 - 5	Constant	1.00	0.00	0.00
	Noisy	0.00	1.00	0.00
	None	0.79	0.21	0.00
5 - 25	Constant	1.00	0.00	0.00
	Noisy	1.00	0.00	0.00
	None	1.00	0.00	0.00
25 - 100	Constant	1.00	0.00	0.00
	Noisy	1.00	0.00	0.00
	None	1.00	0.00	0.00
100 - 300	Constant	0.80	0.07	0.13
	Noisy	0.92*	0.08*	0.00*
	None	0.87	0.07	0.07
300 +	Constant	1.00	0.00	0.00
	Noisy	0.92*	0.08*	0.00*
	None	1.00	0.00	0.00
No Latency	Constant	0.96	0.02	0.02
	Noisy	0.92	0.08	0.00
	None	0.95	0.04	0.01

Table A.5: Probability masses used for the Mode Detector for Dempster-Shafer, Averaging, and Cumulative Belief Fusions

Latency (ms)	Variability	m(DetSig)	m(DetFP)	m(DetAny)
0 - 5	Constant	0.62	0.00	0.38
	Noisy	0.15	0.38	0.46
	None	0.46	0.13	0.41
5 - 25	Constant	0.62	0.00	0.38
	Noisy	0.59	0.00	0.41
	None	0.60	0.00	0.40
25 - 100	Constant	0.85	0.00	0.15
	Noisy	0.90	0.03	0.06
	None	0.88	0.02	0.10
100 - 300	Constant	0.71	0.0	0.29
	Noisy	0.65*	0.08*	0.27*
	None	0.71	0.00	0.29
300 +	Constant	0.44	0.00	0.56
	Noisy	0.65*	0.08*	0.27*
	None	0.44	0.00	0.56
No Latency	Constant	0.61	0.0	0.39
	Noisy	0.65	0.08	0.27
	None	0.62	0.02	0.36

Table A.6: Probability masses used for the HMM Detector for Dempster-Shafer, Averaging, and Cumulative Belief Fusions

Appendix B

Prior Probabilities for Bayes' Theorem

Probability masses marked with an asterisk use the probability samples for the equivalent No Latency category because no samples were available.

Latency (ms)	Variability	P(Sig)	P(!Sig)	P(Det|Sig)	P(Det|!Sig)	P(FP)	P(!FP)	P(Det|FP)	P(Det|!FP)
0 - 5	Constant	0.36	0.67	0.53	0.17	0.06	0.94	0.00	0.32
	Noisy	0.12	0.88	0.00	0.07	0.44	0.56	0.00	0.11
	None	0.29	0.71	0.47	0.13	0.17	0.83	0.00	0.28
5 - 25	Constant	0.47	0.53	0.73	0.08	0.04	0.96	0.00	0.4
	Noisy	0.67	0.33	0.65	0.00	0.02	0.98	0.00	0.44
	None	0.57	0.43	0.68	0.05	0.03	0.97	0.00	0.42
25 - 100	Constant	0.80	0.20	0.58	0.17	0.03	0.97	0.00	0.52
	Noisy	0.82	0.18	0.47	0.20	0.02	0.98	0.00	0.43
	None	0.81	0.19	0.51	0.19	0.02	0.98	0.00	0.46
100 - 300	Constant	0.49	0.51	0.62	0.26	0.02	0.98	0.00	0.45
	Noisy	0.59*	0.41*	0.51*	0.07*	0.13*	0.87*	0.00*	0.38*
	None	0.49	0.51	0.62	0.26	0.20	0.80	0.00	0.45
300 +	Constant	0.34	0.66	0.51	0.44	0.05	0.95	0.00	0.49
	Noisy	0.59*	0.41*	0.51*	0.07*	0.13*	0.87*	0.00*	0.38*
	None	0.34	0.66	0.51	0.44	0.05	0.95	0.00	0.49
No Latency	Constant	0.43	0.57	0.58	0.29	0.04	0.96	0.00	0.44
	Noisy	0.59	0.41	0.51	0.07	0.13	0.87	0.00	0.38
	None	0.47	0.53	0.56	0.25	0.06	0.94	0.00	0.42

Table B.1: Probability values used for the Plateau Detector for Bayes' Theorem

Latency (ms)	Variability	P(Sig)	P(!Sig)	P(Det|Sig)	P(Det|!Sig)	P(FP)	P(!FP)	P(Det|FP)	P(Det|!FP)
0 - 5	Constant	0.36	0.64	0.20	0.15	0.06	0.94	0.00	0.18
	Noisy	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	None	0.29	0.71	0.18	0.10	0.17	0.83	0.00	0.14
5 - 25	Constant	0.47	0.53	0.36	0.56	0.04	0.96	0.50	0.47
	Noisy	0.67	0.33	0.26	0.07	0.02	0.98	0.00	0.20
	None	0.57	0.43	0.30	0.38	0.03	0.97	0.33	0.86
25 - 100	Constant	0.80	0.20	0.33	0.17	0.03	0.97	1.00	0.28
	Noisy	0.82	0.18	0.13	0.10	0.02	0.98	0.00	0.13
	None	0.81	0.19	0.20	0.13	0.02	0.98	0.50	0.18
100 - 300	Constant	0.49	0.51	0.28	0.04	0.02	0.98	0.50	0.17
	Noisy	0.59*	0.41*	0.18*	0.04*	0.13*	0.87*	0.00*	0.14*
	None	0.49	0.51	0.28	0.04	0.02	0.98	0.50	0.17
300+	Constant	0.34	0.66	0.24	0.19	1.00	0.05	0.95	0.17
	Noisy	0.59*	0.41*	0.18*	0.04*	0.13*	0.87*	0.00*	0.14*
	None	0.34	0.66	0.24	0.19	0.05	0.95	1.00	0.17
No Latency	Constant	0.43	0.57	0.27	0.20	0.04	0.96	0.59	0.21
	Noisy	0.59	0.41	0.18	0.04	0.13	0.87	0.00	0.14
	None	0.47	0.53	0.24	0.16	0.06	0.94	0.29	0.20

Table B.2: Probability values used for the Changepoint Detector for Bayes' Theorem

Latency (ms)	Variability	P(Sig)	P(!Sig)	P(Det|Sig)	P(Det|!Sig)	P(FP)	P(!FP)	P(Det|FP)	P(Det|!FP)
0 - 5	Constant	0.36	0.67	0.57	0.08	0.06	0.94	0.00	0.27
	Noisy	0.12	0.88	0.50	0.33	0.44	0.56	0.27	0.42
	None	0.29	0.71	0.56	0.17	0.17	0.83	0.20	0.30
5 - 25	Constant	0.47	0.53	0.64	0.12	0.04	0.96	0.00	0.38
	Noisy	0.67	0.33	0.65	0.53	0.02	0.98	0.00	0.62
	None	0.57	0.43	0.64	0.28	0.03	0.97	0.00	0.50
25 - 100	Constant	0.80	0.20	0.75	0.67	0.03	0.97	0.00	0.76
	Noisy	0.82	0.18	0.69	0.60	0.02	0.98	0.00	0.69
	None	0.81	0.19	0.71	0.63	0.02	0.98	0.00	0.71
100 - 300	Constant	0.49	0.51	0.80	0.87	0.02	0.98	1.00	0.83
	Noisy	0.59*	0.41*	0.66*	0.44*	0.13*	0.87*	0.24*	0.62*
	None	0.49	0.51	0.80	0.87	0.02	0.98	1.00	0.83
300 +	Constant	0.34	0.66	0.73	0.56	0.05	0.95	0.00	0.65
	Noisy	0.59*	0.41*	0.66*	0.44*	0.13*	0.87*	0.24*	0.62*
	None	0.34	0.67	0.73	0.56	0.05	0.95	0.00	0.65
No Latency	Constant	0.43	0.57	0.71	0.47	0.04	0.96	0.12	0.60
	Noisy	0.59	0.41	0.66	0.44	0.13	0.87	0.24	0.62
	None	0.47	0.53	0.70	0.46	0.06	0.94	0.18	0.60

Table B.3: Probability values used for the TEntropy-StdDev Detector for Bayes' Theorem.

Latency (ms)	Variability	P(Sig)	P(!Sig)	P(Det|Sig)	P(Det|!Sig)	P(FP)	P(!FP)	P(Det|FP)	P(Det|!FP)
0 - 5	Constant	0.36	0.67	0.87	0.72	0.06	0.94	1.00	0.76
	Noisy	0.12	0.88	0.00	0.43	0.44	0.56	0.53	0.26
	None	0.29	0.71	0.76	0.61	0.17	0.83	0.65	0.66
5 - 25	Constant	0.47	0.53	0.64	0.24	0.04	0.96	0.00	0.44
	Noisy	0.67	0.33	0.68	0.73	0.02	0.98	1.00	0.69
	None	0.57	0.43	0.66	0.43	0.03	0.97	0.33	0.57
25 - 100	Constant	0.80	0.20	0.71	0.33	0.03	0.97	0.00	0.66
	Noisy	0.82	0.18	0.78	0.30	0.02	0.98	0.00	0.70
	None	0.81	0.19	0.75	0.31	0.02	0.98	0.00	0.69
100 - 300	Constant	0.49	0.51	0.32	0.06	0.02	0.98	0.00	0.19
	Noisy	0.59*	0.41*	0.70*	0.49*	0.13*	0.87*	0.53*	0.63*
	None	0.49	0.51	0.32	0.06	0.02	0.98	0.00	0.18
300 +	Constant	0.34	0.66	0.44	0.01	0.05	0.95	0.00	0.17
	Noisy	0.59*	0.41*	0.70*	0.49*	0.13*	0.87*	0.53*	0.63*
	None	0.34	0.66	0.44	0.01	0.05	0.95	0.00	0.17
No Latency	Constant	0.43	0.57	0.54	0.22	0.04	0.96	0.29	0.36
	Noisy	0.59	0.41	0.70	0.49	0.13	0.87	0.53	0.63
	None	0.47	0.53	0.59	0.28	0.06	0.94	0.41	0.43

Table B.4: Probability values used for the TEntropy-MeanDiff Detector for Bayes' Theorem

Latency (ms)	Variability	P(Sig)	P(!Sig)	P(Det|Sig)	P(Det|!Sig)	P(FP)	P(!FP)	P(Det|FP)	P(Det|!FP)
0 - 5	Constant	0.36	0.64	0.37	0.00	0.06	0.94	0.00	0.14
	Noisy	0.18	0.82	0.00	0.10	0.44	0.56	0.20	0.00
	None	0.29	0.71	0.32	0.04	0.17	0.83	0.15	0.11
5 - 25	Constant	0.47	0.53	0.68	0.00	0.04	0.96	0.00	0.33
	Noisy	0.67	0.02	0.55	0.00	0.02	0.98	0.00	0.38
	None	0.57	0.43	0.60	0.00	0.03	0.97	0.00	0.36
25 - 100	Constant	0.80	0.20	0.42	0.00	0.03	0.97	0.00	0.34
	Noisy	0.82	0.18	0.36	0.00	0.02	0.98	0.00	0.30
	None	0.81	0.19	0.38	0.00	0.02	0.98	0.00	0.31
100 - 300	Constant	0.49	0.51	0.26	0.04	0.02	0.98	0.50	0.14
	Noisy	0.59*	0.41*	0.41*	0.05*	0.13*	0.87*	0.18*	0.28*
	None	0.49	0.51	0.26	0.04	0.02	0.98	0.50	0.14
300 +	Constant	0.34	0.66	0.13	0.00	0.05	0.95	0.00	0.05
	Noisy	0.59*	0.41*	0.41*	0.05*	0.13*	0.87*	0.18*	0.28*
	None	0.34	0.66	0.73	0.56	0.05	0.95	0.00	0.65
No Latency	Constant	0.43	0.57	0.32	0.01	0.04	0.96	0.06	0.15
	Noisy	0.59	0.41	0.41	0.05	0.13	0.87	0.18	0.28
	None	0.47	0.53	0.35	0.02	0.06	0.94	0.12	0.18

Table B.5: Probability values used for the Mode Detector for Bayes' Theorem

Latency (ms)	Variability	P(Sig)	P(!Sig)	P(Det|Sig)	P(Det|!Sig)	P(FP)	P(!FP)	P(Det|FP)	P(Det|!FP)
0 - 5	Constant	0.36	0.64	0.53	0.19	0.06	0.94	0.00	0.33
	Noisy	0.12	0.88	0.50	0.37	0.44	0.56	0.33	0.42
	None	0.29	0.71	0.53	0.25	0.17	0.83	0.25	0.35
5 - 25	Constant	0.47	0.53	0.73	0.40	0.04	0.96	0.00	0.58
	Noisy	0.67	0.33	0.52	0.73	0.02	0.98	0.00	0.60
	None	0.57	0.43	0.60	0.53	0.03	0.97	0.00	0.59
25 - 100	Constant	0.80	0.20	0.71	0.50	0.03	0.97	0.00	0.69
	Noisy	0.82	0.18	0.62	0.30	0.02	0.98	1.00	0.56
	None	0.81	0.19	0.65	0.38	0.02	0.98	0.50	0.60
100 - 300	Constant	0.49	0.51	0.58	0.23	0.02	0.98	0.00	0.41
	Noisy	0.59*	0.41*	0.58*	0.45*	0.13*	0.87*	0.35*	0.55*
	None	0.49	0.51	0.58	0.23	0.02	0.98	0.00	0.41
300 +	Constant	0.34	0.66	0.56	0.36	0.05	0.95	0.00	0.45
	Noisy	0.59*	0.41*	0.58*	0.45*	0.13*	0.87*	0.35*	0.55*
	None	0.34	0.66	0.56	0.36	0.05	0.95	0.00	0.45
No Latency	Constant	0.43	0.57	0.60	0.30	0.04	0.96	0.00	0.45
	Noisy	0.59	0.41	0.58	0.45	0.13	0.87	0.35	0.55
	None	0.47	0.53	0.59	0.33	0.06	0.94	0.18	0.47

Table B.6: Probability values used for the HMM Detector for Bayes' Theorem