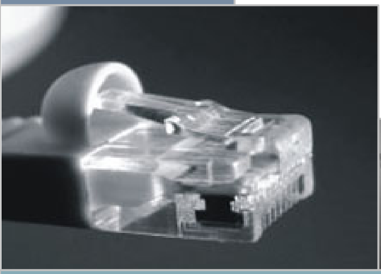
Network Architectures
and Services
NET 2010-06-1

Dissertation

Traffic Anomaly Detection and Cause Identification Using Flow-Level Measurements

Gerhard Münz

Network Architectures and Services
Department of Computer Science
Technische Universität München



TECHNISCHE UNIVERSITÄT MÜNCHEN
Institut für Informatik
Lehrstuhl für Netzarchitekturen und Netzdienste

Traffic Anomaly Detection and Cause Identification Using Flow-Level Measurements

Gerhard Münz

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktor-Ingenieurs (Dr.-Ing.)

genehmigten Dissertation.

Die Dissertation wurde am 29.09.2009 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 12.05.2010 angenommen.

Cataloging-in-Publication Data
Gerhard Münz
Traffic Anomaly Detection and Cause Identification Using Flow-Level Measurements
Dissertation, June 2010
Network Architectures and Services, Department of Computer Science
Technische Universität München

ISBN: 3-937201-12-2
ISSN 1868-2634 (print)
ISSN 1868-2642 (electronic)
Network Architectures and Services NET 2010-06-1
Series Editor: Georg Carle, Technische Universität München, Germany

ABSTRACT

Measurement and analysis of traffic in IP networks are of great interest for network operators as they provide important information about the utilization of network resources, the user behavior, as well as the deployed applications and services. In particular, flow-level traffic measurements have become more and more ubiquitous in the past years. In this context, a flow is defined as a stream of packets which are observed at a given interface in the network and which share a set of common properties called flow keys. For each flow, a flow record is generated containing the flow keys as well as additional flow attributes and statistical information, such as the observation time of the flow, the number of bytes and packets etc.

This dissertation deals with the detection of traffic anomalies and the identification of their causes using flow-level measurement data. Traffic anomalies are significant deviations from the pattern of usual network traffic. Possible reasons for traffic anomalies are changes in the network topology (e.g., newly connected hosts, routing changes) or network usage (e.g., changed customer behavior, new applications). Anomalies may also be caused by failures of network devices as well as by malicious worm or attack traffic. The early detection of such events is of particular interest as they may impair the safe and reliable operation of the network.

For the detection of traffic anomalies, we convert the flow records into time series of eight different traffic metrics describing the traffic volume as well as the cardinality of certain flow keys. We investigate various statistical change detection methods which have been originally conceived for quality control in manufacturing processes. In particular, we use control charts to detect shifts in the mean, standard deviation, or correlation of traffic measurement variables. As most traffic measurement variables exhibit non-stationarity and serial correlation, residual generation methods need to be applied in order to reduce the effect of systematic changes, such as seasonal variation. For anomaly detection in a single traffic metric, we examine different time-series analysis methods with special focus on robust forecasting techniques. For multi-metric anomaly detection, we study the application of principal component analysis (PCA) which allows modeling the correlation structure between different measurement variables.

The investigated change detection and residual generation methods are evaluated and compared based on flow data which was collected in the network of an Internet service provider (ISP). The detected anomalies are classified according to their relevance from the point of view of the network administrator. The evaluation shows that the combination of exponential smoothing and Shewhart control chart is appropriate for detecting certain kinds of relevant anomalies without requiring much adaptation or parameter tuning. Multi-metric anomaly detection using batch-mode PCA and T^2 control chart yields a very large proportion of relevant alarms, yet at the cost of increased complexity and with a need for training data. In this context, robust M-estimators are useful to reduce the bias caused by outliers in the training data. As an important result, the choice of traffic metrics and the analyzed part of traffic turn out to have a large influence on which kinds of anomalies can be detected.

Just like most traffic anomaly detection approaches, the presented methods are limited to the detection of traffic anomalies but do not provide any information about their causes. Without such information, however, anomaly notifications are quite useless for the network administrator because he cannot assess the relevance of the reported events, nor can he decide on any appropriate reaction. Conducting a manual investigation of the original flow records allows identifying the responsible flows in most cases, yet this detective work is very time-consuming and therefore cannot be afforded. As a solution to this problem, we present a couple of algorithms enabling the automated identification of frequent anomaly causes which are of potential interest for the network administrator, such as scanning activities and brute-force password guessing. With these algorithms, most of the relevant anomalies can be examined without requiring any manual intervention.

Altogether, the methods and algorithms presented in this dissertation provide a feasible solution to detect traffic anomalies and to identify their causes. In addition, they can be easily deployed in high-speed networks.

CONTENTS

Part I

INTRODUCTION AND BACKGROUND

1. INTRODUCTION

1.1 Motivating the Need for Traffic Measurements and Analysis

In the last two decades, the Internet has evolved into a global communication and service network large parts of today's social and economic life depend on. Compared to earlier global telecommunication networks for voice calls and data services, such as ISDN (Integrated Services Digital Network), the Internet is much more open and heterogeneous in many aspects.

With respect to the **organization**, the Internet consists of more than 30.000 autonomous systems[1] which are interconnected to provide end-to-end reachability of all Internet users. Only a small area of the Internet is subject to centralized control, such as the assignment of domain names and numbers by the Internet Corporation for Assigned Names and Numbers (ICANN). Apart from that, autonomous system operators are independent and maintain bi- or multi-lateral relationships in contracts or peering agreements. The lack of centralized control and the openness of the Internet has been the basis of its fast evolution and enormous growth. In contrast, the organization of telecommunication networks is being regulated by the International Telecommunication Union (ITU) since 1865, which does not only enact standards but also decides on the future development of the network.

Regarding **technology**, the Internet realizes a unified packet switched overlay network on top of a large variety of communication technologies, including wired as well as wireless networks. This is illustrated in the two bottom layers of the hourglass shown in Figure 1.1. The characteristics of these underlying technologies may be very different, covering a large spectrum from very slow to very high-speed links, and short to long transmission delays. Furthermore, there are large mutual dependencies between different data streams due to the statistical multiplexing of packets. The division of the address space into many hierarchical subnets and the usage of distributed routing algorithms makes the Internet a very dynamic and complex system. Earlier telecommunication networks are mostly circuit switched with a sophisticated signaling infrastructure and a widely centralized control plane which controls the switching of channels between users.

As shown in the upper half of Figure 1.1, the Internet is also characterized

[1] http://www.cidr-report.org/

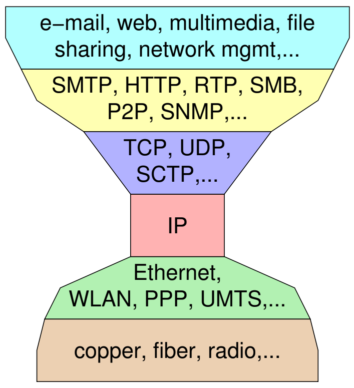

e–mail, web, multimedia, file sharing, network mgmt,...
SMTP, HTTP, RTP, SMB, P2P, SNMP,...
TCP, UDP, SCTP,...
IP
Ethernet, WLAN, PPP, UMTS,...
copper, fiber, radio,...

Fig. 1.1: Hourglass of the Internet protocol layers

by an ever growing variety of **protocols and services** running on top of the Internet Protocol (IP), such as e-mail, world wide web, file sharing, multimedia, or telephony. The separation between pure data transport provided by the "stupid network" [Ise97] and the services offered by the applications hosted at the end systems facilitates the launch of new Internet services and protocols. On the other hand, running all these different services over the same network is a big challenge because of the various quality of service demands and traffic characteristics of the different services. Telecommunication networks, in contrast, have been designed for specific and previously known services, which made it possible to optimize the network for the given purpose.

Finally, heterogeneity and openness also applies to the **users and utilization** of the Internet. It is used for human-to-human, human-to-machine, and machine-to-machine communication. Internet users are private customers, professionals, and academics, with noncommercial or commercial interests. Criminals exploit the openness of the Internet for concealed communication and for committing crimes, such as computer fraud and espionage.

As a major consequence of the Internet's openness and heterogeneity, network operators lose control on the utilization of their networks. Since they do not know exactly for which kinds of applications and services the network is used, the traffic is far less predictable than in telecommunication networks. In addition, it is very difficult to prevent harmful or unwanted traffic from entering the network. Therefore, traffic measurements have become an essential source of information in the monitoring infrastructure of the Internet. In particular, flow-level traffic measurements are now commonly deployed in the backbone networks of Internet service providers.

Topic of this dissertation is the analysis of flow-level traffic measurement data for real-time detection of significant changes in selected traffic metrics. Such changes can be signs of traffic anomalies that affect the operation of the network. Traffic anomaly detection has been a vivid area of research for several years, especially in the context of network intrusion detection systems. However, the anomaly detection approach suffers from inherent limitations which often inhibit the practical deployment. In the following section, we discuss these problems and set up a list of criteria which should be reflected in the search for applicable traffic anomaly detection methods. Thereafter, Section 1.3 summarizes the challenges of using statistical change detection methods for traffic anomaly detection. Finally, Section 1.4 states the main contribution of our work and explains the document structure.

1.2 A Reflective Approach to Traffic Anomaly Detection

Research on traffic anomaly detection is based on two main assumptions. The first assumption is that network traffic shows certain characteristics under normal conditions which can be described somehow, typically by means of a model. This model is sometimes called ‘baseline’, especially if it refers to variables which are monitored over time. The model parameters either are known a-priori or can be learned by observation. The second assumption is that deviations from this normal behavior are rare and expected to be the result of events that are of potential interest for the network administrator.

Beyond these assumptions, traffic anomaly detection is frequently presented in the literature as a method to detect malicious activities, such as attacks, spreading worms, or botnet communication. A good overview on such anomaly-based network intrusion detection approaches until the year 2004 is presented in a survey paper by Estevez et al. [ETGTDV04a]. The hypothesis behind these approaches is that traffic anomalies are commonly the result of malicious activities. Hence, anomaly detection is seen as an alternative to the so-called ‘misuse detection’ approach which aims for the identification of known patterns of harmful traffic. The main advantage repeatedly brought forward by the authors is that anomaly detection allows discovering new and unknown malicious activities whereas misuse detection can only recognize previously known attacks.

Equalizing traffic anomalies with malicious traffic is, however, a problematic guess, not to say speculation. There are many other causes for traffic anomalies, such as spontaneous changes in the user behavior or a newly deployed server or application. On the other hand, malicious traffic may resemble non-malicious traffic such that the two cannot be distinguished.

Gates and Taylor question the application of anomaly detection to network-based intrusion detection in a “provocative discussion” [GT06]. In addition to the already mentioned disputable base assumptions, they argue that it is difficult to ensure that training data used to characterize non-malicious traffic is free of attacks and, at the same time, representative for the entirety of normal traffic variations. Furthermore, they stress that operational usability requires that no or very little manual intervention of the network administrator is necessary. Alarms must indicate an event of interest in a reliable way, meaning that false alarms, even if occurring with low probability, are often unacceptable. Also, administrators usually cannot spend a lot of time on fine-tuning many different parameters until the detection method provides reasonable results.

A general resort is to generalize the scope of traffic anomaly detection. Apart from the security incidents discussed above, traffic anomalies caused by failures of network devices or significant changes in the user behavior are then accounted as events of interest. Indeed, the detection of such events can be useful for the network administrator as long as it is possible to distinguish different anomaly causes. Ideally, the detection method identifies those parts of traffic which are responsible for the anomaly, as well as the involved hosts. Many anomaly detection methods, however, do not provide such information, meaning that the measurement data must be inspected manually in order to identify the anomaly causes.

Several traffic anomaly detection approaches make use of very sophisticated statistical methods (e.g., [SST05, LCD05, SBK08]). Referring to evaluations performed with synthetic or real traffic traces, the authors typically claim that their methods yield excellent results. However, sophisticated statistical methods often depend on many parameters which require fine-tuning until satisfactory detection results can be obtained. Therefore, the question arises if similar results could have been achieved with simpler methods which are less complex and easier to parameterize. Furthermore, it seems that research has placed too much emphasis on statistical methods whereas only few publications have concentrated on the selection of appropriate traffic metrics to be monitored [KR06, BMP07] although this must be considered as an equally important problem.

As a result of these considerations, we formulate the following **evaluation criteria** for our own work on traffic anomaly detection:

Ease of deployment: Detection methods should yield useful results without requiring elaborate fine-tuning. In other words, the method should be robust with respect to suboptimal parametrization.

Justification of complex methods: Complex methods are more costly to implement and deploy. Therefore, simple methods should be preferred over more complex ones unless the increased complexity can be justified by superior detection results.

Relevance of the detected anomalies: The statistical significance of an anomaly is not always tantamount to its importance for the network administrator. Therefore, the relevance of the detection results should be critically reflected in the evaluation.

Identification of anomaly causes: As far as possible, the cause of an anomaly should be characterized and identified in an automated way to minimize the need for manual inspection of the measurement data.

As an outcome of this dissertation, we will show that these criteria can be fulfilled in a satisfactory manner by combining statistical change detection methods and algorithms which enable the identification of common anomaly causes. In the next section, we give an introduction to statistical process control where change detection methods have first been deployed.

1.3 Change Detection by Means of Statistical Process Control

The idea of **statistical process control** (SPC) is to monitor specific output variables of a system and to raise an alarm if the observed values run out of the range of sound operation. The purpose is the detection of unexpected changes in characteristic properties of the system because such changes may be indications of failures, malfunctions, and wearout. SPC was originally developed and deployed as part of quality control to monitor manufacturing processes [She31]. In such processes, deviations from the norm likely deteriorate the quality of the finished products and thus increase scrap or the need of rework. Therefore, variables exceeding the margins defining the 'in-control state' of the process are to be detected quickly in order to enable manual intervention, recalibration, or exchange of erroneous components to minimize loss or damage.

Process variables usually exhibit a certain variation under normal conditions which goes back to 'common causes of variation'. Hence, the challenge of SPC is to distinguish this variation from unusual changes caused by events of significance (or 'special causes of variation'). SPC solves this with help of statistical methods. The main tools are so-called **control charts** which are graphical representations of statistical change detection algorithms.

Control charts monitor the mean, the variance, or both, of a time series which ideally resembles 'white noise' under normal conditions (i.e., the output of independent and identically distributed random variables). If the process variable to be monitored does not fulfill this condition, the direct application of control charts does not lead to reasonable results. However, regular variation and correlation in the original variables can be modeled and eliminated with help of appropriate statistical **residual generation** techniques. Examples are time-series analysis, which models temporal dependencies, and principal component analysis, which models correlation between multiple process variables. We then can apply control charts to the residuals, which contain the remaining variability not unexplained by the model.

The need to distinguish regular variation from significant changes is not limited to the monitoring of manufacturing processes. Researchers have attempted to adopt SPC methods for other problems as well. One example is the temporal partitioning of signals into homogeneous segments, which can be useful for speech recognition in audio signals [BB83]. Obviously, there is also a strong relationship to the problem of traffic anomaly detection where significant changes in the statistical properties of the measured traffic are considered as potential signs of network failures or abuses. However, this implies that appropriate residual generation methods can be found because network traffic, just like many process variables, does not resemble white noise.

In this dissertation, we examine the applicability of different residual generation techniques and different control charts in order to detect traffic anomalies in traffic measurement time series. The traffic metrics considered in this work are derived from flow records exported by routers, switches, or network monitors via IPFIX (IP Flow Information Export) [CBS+08], *Cisco NetFlow* [CSVD04], or alternative export mechanisms. The advantage of analyzing traffic at the level of time series instead of flows is that less computational and memory resources are required. As the complexity is constant per time interval and does not depend on the traffic volume or composition, the approach is very well adapted for anomaly detection in high-speed networks.

1.4 Contribution and Document Structure

The **contribution** of this dissertation is the presentation and evaluation of simple and effective methods for the detection of anomalies in traffic measurement time series as well as the identification of their causes. For this purpose, we compare various single-metric and multi-metric anomaly detection approaches based on time-series analysis and principal component analysis (PCA).

Among the considered single-metric techniques, we identify the combination of robust time-series forecasting with the so-called Shewhart control chart as most effective anomaly detection method yielding a large proportion of relevant alarms when applied to appropriate traffic metrics [MC08a]. This method is easy to parameterize and to deploy and thus fulfills two of the criteria listed in Section 1.2. Multi-metric anomaly detection based on PCA has the advantage that we can monitor all metrics with a single T^2 or T_H^2 control chart. On the other hand, PCA is more complex and requires appropriate training data to estimate the correlation between different traffic metrics.

In order to obtain reproducible and comparable results, the evaluation of the presented anomaly detection methods is performed offline with flow data collected in the network of an Internet service provider (ISP). However, an important property of the analyzed methods is that they are suitable for traffic anomaly detection in near real-time. For example, some of the methods

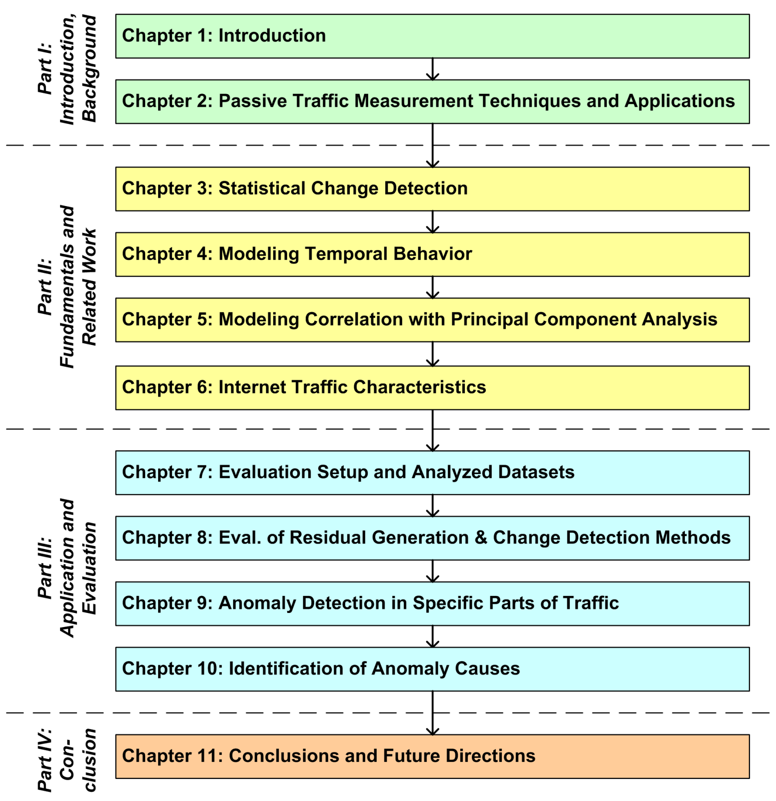

Part I: Introduction, Background
Chapter 1: Introduction
Chapter 2: Passive Traffic Measurement Techniques and Applications
Part II: Fundamentals and Related Work
Chapter 3: Statistical Change Detection
Chapter 4: Modeling Temporal Behavior
Chapter 5: Modeling Correlation with Principal Component Analysis
Chapter 6: Internet Traffic Characteristics
Part III: Application and Evaluation
Chapter 7: Evaluation Setup and Analyzed Datasets
Chapter 8: Eval. of Residual Generation & Change Detection Methods
Chapter 9: Anomaly Detection in Specific Parts of Traffic
Chapter 10: Identification of Anomaly Causes
Part IV: Con-clusion
Chapter 11: Conclusions and Future Directions

Fig. 1.2: Chapter guide through this document

have already been integrated as detection modules into our traffic analysis framework *TOPAS* [BM06, MC07]. The required flow data can be obtained from commercial routers or with help of open-source implementations of monitoring devices, such as our monitoring toolkit *VERMONT* [LSMD06]

In order to assess the relevance of the detected anomalies and to identify their causes, we present algorithms which search the original flow data for traffic patterns of known kinds of incidents. These algorithms make a manual inspection of the data unnecessary for most of the alarms and therefore are essential for the operational deployment of the anomaly detection methods.

Figure 1.2 illustrates the chapter **structure of this document**. As can be seen, the chapters are grouped in four parts: "Introduction and Background", "Fundamentals and Related Work", "Application and Evaluation", and "Conclusion". In the following, we briefly summarize the content of each chapter.

In **Chapter 2**, we begin with an overview on passive traffic measurement techniques and their applications. In particular, we describe the different levels of granularity, from link level to packet level, at which passive traffic measurements can be performed. We show how more detailed information about the traffic increases the possibilities of traffic analysis, yet at the cost of increasing the computation and memory requirements. Chapter 2 also summarizes the current state of IETF standardization regarding the measurement and export of flow information, to which we have been actively contributing for a couple of years [MCA10, CAJM10, DKCM10]. The section terminates with a presentation of tools for traffic measurement [MADC06, LSMD06, DM06, BMC10] and analysis [BM06, MC07, MLC07, MWC07, MC08b] which we have developed in the context of various research projects (e.g., [MFC^{+}05]).

Part II of the document presents the statistical fundamentals of change detection, time-series analysis, and principal component analysis. These methods form the base of our traffic anomaly detection approach.

In **Chapter 3**, we give an overview on statistical change detection and explain different types of control charts. Thereby, we elaborate the underlying assumptions and prerequisites since a thorough understanding of the fundamentals is necessary to deploy these statistical methods appropriately. Furthermore, we discuss existing traffic anomaly detection approaches which make use of these methods.

Chapter 4 summarizes the basic concepts and methods of time-series analysis. We are particularly interested in forecasting techniques which can be used for residual generation. We find that exponential smoothing and Holt-Winters forecasting are appropriate for our purposes because they are quite robust against uncertainties regarding the distribution of the observed variables. As in the previous chapter, we combine the presentation of the fundamentals with a discussion of related work deploying time-series analysis methods for traffic anomaly detection.

While time-series analysis considers temporal dependencies, PCA examines the correlation structure between different variables. **Chapter 5** explains how to determine the principal component transformation for a given data set. We give an overview on PCA-based statistical methods and significance tests and discuss specific aspects which are related to the application of PCA to multivariate time series. Furthermore, we dedicate one section to existing work on traffic anomaly detection using PCA.

In the 1990s, it was discovered that Internet traffic in local-area and wide-area networks exhibits self-similar structures and long-range dependence in certain traffic characteristics. Chapter 6 gives an overview on the properties of self-similar traffic as well as possible explanations for its existence. Furthermore, we assess the effect of long-range dependence on time-series forecasting and anomaly detection by reviewing results reported in the literature.

Part III of this document covers the application and evaluation of the presented methods for traffic anomaly detection. At the beginning, **Chapter 7** describes the flow dataset our evaluation relies on. Furthermore, we explain how the flow records are converted into time series. The considered traffic metrics characterize volume as well as distributional properties of the traffic.

In **Chapter 8**, we apply several combinations of the residual generation techniques and change detection methods presented in Part II to time series derived from the overall IP traffic. By inspecting the original flow data, we identify the reason for every detected anomaly and assess the relevance of the alarm for the network administrator. Furthermore, we study the influence of the monitored traffic metric on the detection results.

Chapter 9 deals with the detection of anomalies in specific parts of the traffic which we assume to be affected by incidents of interest. For example, we apply change detection methods to time series of ICMP traffic. The results show that the proportion of relevant anomalies is higher compared to the analysis of the overall traffic.

The considered anomaly detection methods are based on traffic measurement time series and therefore cannot provide any information about the responsible flows or the cause of an anomaly. To fill this gap, we present three algorithms in **Chapter 10** which search the flow records for known traffic patterns of incidents which occur quite frequently and which are of potential interest for the network administrator, namely network scans, port scans, and password guessing. For these incidents, the algorithms deliver a description of the involved flows and hosts in an automated way.

Part IV concludes the dissertation with a summary of results, an evaluation of achievements, and an outlook to future research in **Chapter 11**.

2. PASSIVE TRAFFIC MEASUREMENT TECHNIQUES AND APPLICATIONS

2.1 Introduction

Traffic measurement methods can be categorized into active and passive techniques [Ziv06]. **Active measurements** are performed with probe packets which are injected into the network. Typically, the injected packets are sent from one end-system to another, which enables the measurement of end-to-end delay. Another relevant application of active measurements is the estimation of path capacities and available bandwidth by sending pairs or trains of packets. Since active measurements are intrusive, they risk to bias the measurement result and to disturb the existing traffic. On the other hand, they do not require access to internal network elements.

In contrast, **passive measurements** are non-intrusive and rely on the observation of existing traffic without changing or manipulating its characteristics. Passive measurements can be performed on end-systems to monitor the traffic at local interfaces. In order to measure the traffic going through a network, access to one or multiple observation points, such as switches, routers, gateways, or firewalls, is required. The traffic can be measured at different levels of granularity, such as packet level, flow level, or link level, depending on the measurement technology and the usage of the measurement results.

As our work is based on passive traffic measurements, the next section explains the different passive measurement techniques in more detail. We also discuss the aspects of computational and memory resources which are required to perform these measurements. Thereafter, Section 2.3 gives an overview on applications that make use of data from passive measurements. Relevant Internet standards as well as monitoring tools developed by the computer networking group at the University of Tübingen and later at the Technische Universität München are presented in Section 2.4. Section 2.5 concludes this chapter with the positioning of this dissertation in the given context.

2.2 Passive Measurement Techniques

In this section, we explain and compare packet-level, flow-level, and link-level measurement techniques. Furthermore, we discuss the required resources and how these measurements can be practically performed by monitoring devices with resource restrictions.

2.2.1 Packet-Level Measurements

Passive traffic measurements at packet-level gather information about individual packets. Packet-level measurements are performed by network analyzers and network-based intrusion detection systems which analyze the captured packets directly. Alternatively, packet capturing and analysis can take place at different locations. For example, packets can be recorded at routers, switches, and network monitors from which the resulting measurement data is transported to a remote analysis systems. One standardized option is to implement the Remote Network Monitoring (RMON) MIB module [Wal00] which enables the storage of captured packets matching a given filter. A remote analyzer can then retrieve the packet data with help of the Simple Network Management Protocol (SNMP) [Pre02]. Another standard solution is to export packet reports to a remote collector using the IPFIX (IP Flow Information eXport) protocol [CJQ09].

For some applications, such as session analysis and signature detection, entire packets including packet payload have to be captured. For other applications, such as one-way-delay measurements, it is sufficient to capture packet headers. References to application examples will be provided later in Section 2.3.

The processing and storage resources required for packet-level measurements increase with the number of captured packets. Systems that are capable of monitoring every packet on a high-speed link are very expensive. To cope with this problem, the measurement can be restricted to a subset of the observed packets by applying **filters** or **sampling algorithms**. Filters select packets depending on specific packet properties, such as values of packet header fields. Sampling algorithms, in general, aim at choosing a subset of packets which allows inferring statistics of the entirety of all packets, such as the frequency of a specific packet property. If packet sampling is combined with payload analysis, the goal is to omit packets where we do not expect to find any interesting content, and to keep all packets of potential interest. For this purpose, we have developed a sampling algorithm which deploys Bloom filters to select the first packets of every TCP connection, and have shown that these packets are sufficient for detecting many kinds of worm and botnet activities [BMC10].

Sampling algorithms can be random or systematic. A random sampling algorithm yields a non-deterministic packet selection that depends on a random process. In contrast, the packet selection of a systematic sampling algorithm is deterministic. A widely used systematic sampling method is count-based sampling which selects the first n packets out of each series of $N > n$ packets.

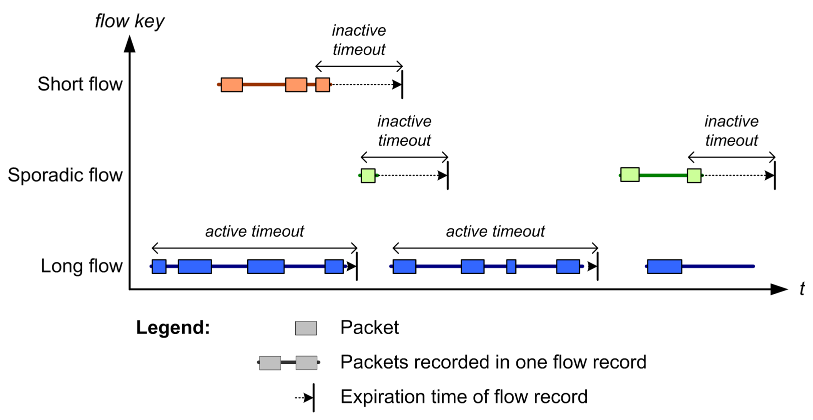


Fig. 2.1: Flow-level measurements with timeouts

A couple of filtering and sampling techniques have been standardized in RFC 5475 [ZMD+09] by the PSAMP (Packet SAMPling) working group at the IETF (Internet Engineering Task Force). RFC 5474 [DCC+09] discusses the complexity of different packet selection methods and sketches various use cases. More information about the PSAMP standardization is given below in Section 2.4.1.

2.2.2 Flow-Level Measurements

Flow-level measurements do not collect information about individual packets but entire flows of packets. According to the definition of the IPFIX (IP Flow Information eXport) working group at the IETF, a flow is defined as a unidirectional stream of IP packets which are observed at an observation point in the network and share a set of common properties called **'flow keys'** [CBS+08]. A usual flow key set is defined by the IP quintuple of transport protocol, source IP address, destination IP address, source port, and destination port. Practical implementations, such as Cisco's *NetFlow* [Cis10b] and Juniper's *J-Flow* [Jun10], use the type of service (ToS) field and the input interface as two additional flow keys. Recent products and implementations allow the user to configure the flow keys in a flexible way. Examples are the most recent *NetFlow* technology called *Flexible NetFlow* [Cis10a] and the open-source monitoring probe *VERMONT* [LSMD06] which will be presented in more detail in Section 2.4.2.

For every observed flow, the metering process determines a couple of statistics, such as the observation time of the first and last packet, and the number of packets and bytes that belong to the flow. The flow keys and the flow statistics are stored in a flow record. Monitoring devices are usually not designed to process, analyze, or archive flow records. Instead, the flow records are exported to a remote system called 'collector'. The export is usually triggered by two flow expiration timeouts as illustrated in Figure 2.1. The **passive timeout** defines a time interval of inactivity (no packets observed) after which a flow record is exported. In addition, flow records of long-lasting flows are periodically exported after the **active timeout**.

Many routers and switches allow the activation of flow-level measurements on one or multiple of their interfaces. Industry consortia and device manufacturers have developed and standardized various protocols for exporting flow records to collectors, such as the *sFlow* protocol [PPM01] and different versions of the *NetFlow* protocol (e.g., [CSVD04]). In the near future, most of the vendors will likely support the IPFIX protocol [CBS$^+$08], which is a recent Internet standard of the IETF. More information about IPFIX is given in Section 2.4.1.

The resource consumption of flow-level measurements depends on the number of monitored packets and the number of flow records that are simultaneously stored in the monitoring device. The number of monitored packets influences the computational load since every packet has to be classified to a new or existing flow record. The number of stored flow records relates to the required amount of memory.

2.2.3 Link-Level Measurements

Link-level measurements are omnipresent as they are performed by nearly every network interface card driver. The simplest link-level measurement counts the number of packets and bytes leaving from and arriving at a given interface. In this case, the resource consumption is negligible as only four counter variables are needed per interface. More sophisticated measurements keep separate counters for different protocols.

The measurement data is typically stored in the MIB from which it can be queried via SNMP. For example, the Interfaces Group MIB module (IF-MIB) [MK00] includes counters for the entire number of packets and bytes that have passed an interface since the system start. More detailed traffic statistics can be stored in the Remote Network Monitoring MIB modules (RMON-MIB, RMON2-MIB) [Wal00, Wal06].

2.2.4 Coping with Resource Restrictions

As we have already mentioned in the preceding subsections, the resources required by passive traffic measurements mainly depend on the level of granularity. While link-level measurements are almost for free, computational and memory requirements are a significant factor for flow-level and packet-level measurements, especially in high-speed networks. If the monitoring device exports the measurement data to a collector, the required bandwidth also has to be taken into consideration. There are several ways to trade the required resources off against the achieved measurement accuracy, which are discussed in the following.

As mentioned in Section 2.2.1, **packet filtering and sampling** allow us to select a subset of the observed packets. Since both of them reduce the number of monitored packets, they are similarly appropriate to save memory. However, with respect to the computational complexity, there are significant differences. For example, a filter which classifies packets based on packet content (e.g., packet header fields) still requires a certain number of comparison operations per packet. On the other hand, a simple systematic count-based sampler makes its decision based on a counter which is increased with every packet. Thus, the effort is very low.

Packet filtering and sampling can be combined with flow-level measurements. In fact, systematic count-based sampling is commonly deployed in order to reduce the number of packets to be processed by the flow metering process. An implementation example is *Sampled NetFlow* by Cisco. As a consequence, the properties and statistics of the original traffic can only be estimated and some flows, especially short ones, risk not to be recorded at all.

In the case of flow-level measurements, the number of flow records which have to be kept in the memory of the monitoring device heavily depends on the set of flow keys. By using a smaller set of flow keys, the required memory can be reduced. Furthermore, masking or hashing flow key fields to a smaller value range may also result in a smaller number of records. As an example, we can store subnet addresses instead of entire host IP addresses. For this purpose, we proposed a rule-based approach for **flexible flow measurement and aggregation** and evaluated the achievable reduction in the number of flow records for different sets of flow keys as well as masked IP addresses based on real traffic [DM06]. The results show that port numbers, especially the port numbers of TCP clients, contribute most of the variability in the flow keys. Thus, not using ports as flow keys leads to a significantly lower number of flow records. Obviously, using fewer or masked flow key fields decreases the measurement resolution in the flow key space.

An alternative approach to reduce the number of records is **flow sampling** [DLT04]. Here, only selected flow records are exported and analyzed while others are dropped after the measurement. Flow sampling is mainly useful to reduce the number of exported flow records. However, it is less suitable for decreasing the amount of flow records which have to be stored in the monitoring device because the flow selection is applied after the flow has timed out (expired). For an extended overview and comparison of different sampling and aggregation techniques, we refer to Duffield's review paper on this topic [Duf04].

To a certain extend, the **timeouts** of flow metering processes influence the computational, memory, and bandwidth requirements. Small timeouts may lead to a larger number of flow records, which increases the amount of

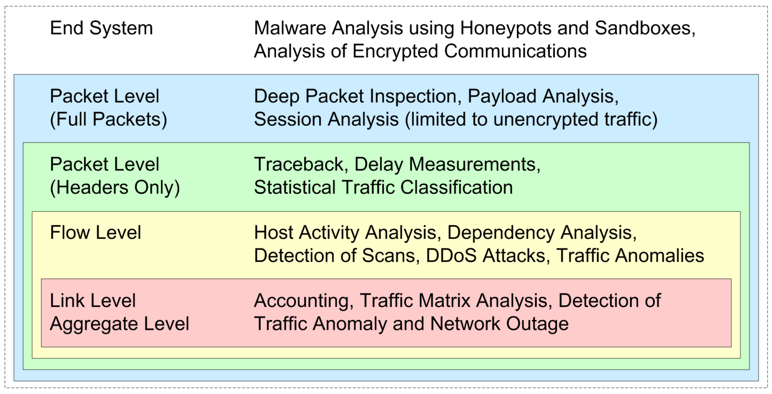

End System
Malware Analysis using Honeypots and Sandboxes, Analysis of Encrypted Communications
Packet Level (Full Packets)
Deep Packet Inspection, Payload Analysis, Session Analysis (limited to unencrypted traffic)
Packet Level (Headers Only)
Traceback, Delay Measurements, Statistical Traffic Classification
Flow Level
Host Activity Analysis, Dependency Analysis, Detection of Scans, DDoS Attacks, Traffic Anomalies
Link Level Aggregate Level
Accounting, Traffic Matrix Analysis, Detection of Traffic Anomaly and Network Outage

Fig. 2.2: Applications of passive traffic measurements

measurement data that has to be exported. It also increases the load on the exporting process because it has to send more records. On the other hand, exporting the flow records more quickly may slightly decrease the average amount of records kept in the memory of the monitoring device.

An economical utilization of the monitoring device's resources can be achieved by adapting the performed traffic measurements as well as the exported information to the application requirements. For example, if only traffic from and to a specific subnet or traffic of a specific transport protocol or port is of interest, an appropriate filter can be configured at the metering process to avoid the measurement of traffic which will not be analyzed later.

In order to facilitate the **management and configuration** of monitoring devices, we developed an XML-encoded configuration data model for IPFIX and PSAMP [MADC06] within the European project *DIADEM Firewall* [DIA06]. The NETCONF protocol [EBC$^+$06] was used to remotely configure instances of our open-source monitoring device implementation *VERMONT*. In 2008, the IPFIX working group at the IETF adopted our concept as the basis of a new Internet standard [MCA10]. More information about the current status of IPFIX standardization, *DIADEM Firewall*, and *VERMONT* are given in Section 2.4.

2.3 Applications of Passive Traffic Measurements

In this section, we give examples for applications which rely on data from passive traffic measurements. In principal, the set of possible applications increases with a finer level of granularity. For example, packet-level measurements allow applying certain traffic analysis methods that are not possible with flow-level measurements. In addition, most of the flow-based applications can be realized as well since flow information can be easily obtained from packet data by aggregation.

The nested boxes in Figure 2.2 illustrate this relationship between measurement granularity and possible applications. The innermost box lists example applications which make use of measurements at the level of links or other traffic aggregates, such as traffic flows between autonomous systems. A classical application is **accounting** traffic volume for billing purposes. In addition, it is possible to determine and examine the traffic matrix of a network with multiple ingress and egress points. Finally, we can detect anomalies and network outages at the link and aggregate level (e.g., [Bru00, BKPR02, LCD04b, SST05, SSRD07]).

The next box contains flow-based applications. In addition to the applications already mentioned for traffic aggregates, flow-level measurements enable the **detection of anomalies** based on flow counts and flow key distribution [LCD05, KR06, SBK08, CPA09]. For example, we have developed an anomaly detection method which applies the k-means clustering algorithm to training data of multiple traffic metrics which can be obtained from flow records [MLC07]. As a result, we obtain different clusters for normal and anomalous traffic. New measurement values can then be classified as normal or anomalous based on the distance to the nearest cluster centroid.

Flow data can also be used to identify worm-infected hosts [DP05] or to detect attacks [DHKR09] with help of heuristics or known attack patterns. Furthermore, the examination of flow data allows analyzing the behavior of individual hosts [GB05] as well as the **relationships and dependencies** between different hosts and services [KPF05, AKM$^+$05, KGE06]. To some extend, the flow statistics make it possible to classify flows into application classes [EBR03]. However, the large majority of **traffic classification** approaches requires information that goes beyond flow data, such as the lengths and arrival times of the packets within a flow.

With packet-level measurements, it is possible to trace the path of a packet along multiple observation points in a network [DG01], which can be used for **one-way delay measurements**. Starting a few years ago, various research groups have begun to use header information of packets belonging to the same unidirectional or bidirectional flow to identify the protocol or application at higher layers [BTS06, KCF$^+$08]. The goal is to classify traffic without considering port numbers and packet payload. For the same purpose, we have developed an efficient traffic classification method which uses Markov models to describe the traffic of different application [DMBC09, MDBC10].

If the traffic measurements provides access to packet payload, **deep packet inspection** is possible, including signature matching algorithms and the analysis of sessions at higher protocol layers [Pax99, Roe99]. Just like aggregates, flows, and packet headers, packet payload can be used for anomaly detection [KTK02, ETGTDV04b]. However, payload analysis is limited to unencrypted traffic.

The outermost box in Figure 2.2 shows the change from traffic measurement to host monitoring. If we had access to one of the end systems of a data transfer, we would be able to analyze traffic before encryption and after decryption. Furthermore, we could study the relationship between traffic and applications, for example by examining the effect of malware on the end system.

2.4 Standards and Tools

At the IETF, two working groups are in close relationship to passive traffic measurements at flow and packet level, namely the IPFIX and the PSAMP working group. The next subsection gives a summary of the standardization activities of these two groups. Thereafter, we present software tools which we have been developing at the Technische Universität München and earlier at the University of Tübingen in the scope of several research projects.

2.4.1 IPFIX and PSAMP Standardization at the IETF

The **IPFIX (IP Flow Information eXport)** working group at the IETF has been developing and standardizing techniques and protocols for flow-level measurements in IP networks and the export of the resulting flow records. Starting in 2001, a long discussion began about the requirements and suitable existing protocols for flow information export. In 2004, the working group published RFC 3917 on the IPFIX requirements [QZCZ04] and RFC 3955 on the evaluation of candidate protocols [Lei04]. In the latter RFC, it was concluded that none of the existing protocols fulfilled the requirements in a satisfying way. Therefore, it was decided to standardize a new protocol based on the most appropriate existing solution, which was *Cisco NetFlow* version 9 [CSVD04].

After this slow start, it took another three and a half years to finally publish the IPFIX protocol specification in RFC 5101 [CBS$^+$08] in 2008. Reasons for the additional delay were discussions about the transport protocol to be chosen. Finally, the Stream Control Transmission Protocol (SCTP) with partial reliability extension [SRX$^+$04] was preferred over UDP, which lacks congestion awareness, and TCP, which only provides fully reliable transport. Hence, RFC 5101 now specifies SCTP as mandatory transport protocol which must be implemented in order to be compliant with the standard; UDP and TCP may be optionally supported as well.

NetFlow version 9 and IPFIX support variable record structures, which allows a monitoring device to export exactly those fields for which measurement data is available. The record structure is defined by a so-called ‘template’. Within a template, the encoding and semantic of each record field is defined by an information element identifier. Common information elements for flow-level measurement data are standardized by the IPFIX information model [QBCM08]. If these are not sufficient, vendors can define and utilize additional enterprise-specific elements.

At the time of writing, we are involved in the standardization of an IPFIX extension which defines a specific mapping between templates and SCTP streams [CAJM10]. The benefit of this approach is that the number of lost records can be calculated per template if SCTP is used with partial reliability. Further IPFIX extensions deal with the export of bidirectional flow data [TB08] and the storage of IPFIX data in files [TBM+09].

Regarding the management and configuration of monitoring devices, a MIB module [DKCM10] has been standardized; an XML configuration data model [MCA10] for usage with the NETCONF protocol [EBC+06] is currently in the final phase of standardization. As mentioned in Section 2.2.4, the initiative to develop and standardize a vendor-independent configuration data model for IPFIX and PSAMP goes back to work we performed in the European project *DIADEM Firewall* [MADC06]. More details about this project are given in the next subsection.

Before its closure in 2009, the **PSAMP (Packet SAMPling)** working group focused on packet selection techniques and the export of the resulting packet data using the IPFIX protocol. Because of the tight linkage to the IPFIX standardization process, the work of the PSAMP group also took a very long time from 2002 till 2009. At the end, three Internet standards were published. Packet filtering and sampling mechanisms are standardized in RFC 5475 [ZMD+09]. RFCs 5476 and 5477 [CJQ09, DCA+09] specify the usage of the IPFIX protocol for PSAMP export as well as necessary extensions of the information model. These standards enable the export of records containing header fields and payload of selected packets.

2.4.2 Tools for Traffic Measurement and Analysis

In this section, we give an overview on traffic measurement and analysis tools which have been created by scientists and students of the computer networking chair at the Technische Universität München (formerly University of Tübingen) and its partners.

In the scope of the *HISTORY* project [DC05], which is a research cooperation with the computer networking group at the University of Erlangen-Nuremberg, we have been developing open-source software tools for distributed traffic measurements and analysis based on the IPFIX and PSAMP standards since 2004. The development has been pushed and financially supported by a couple of national and international projects, such as the European FP6 project *DIADEM Firewall* [DIA06], the *LUPUS* project funded by DFG (German Research Foundation), and the *monk-it* project funded by the German Federal Office for Information Security (BSI). An important outcome of these projects are the open-source software tools *VERMONT* and *TOPAS*.

VERMONT (VERsatile MONnitoring Toolkit) [LSMD06, VER10] is a modular monitoring device implementation which allows capturing packets

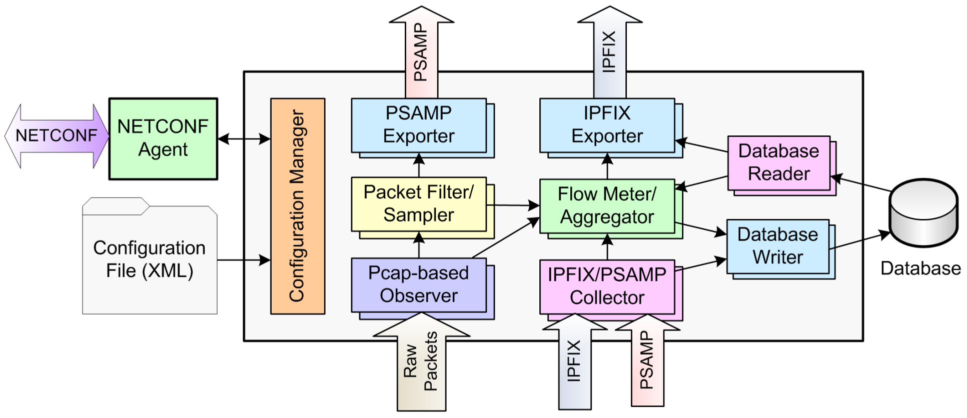


Fig. 2.3: Architecture of *VERMONT*

at interfaces in promiscuous mode in order to generate and export PSAMP packet reports or IPFIX flow records. The architecture of *VERMONT* is depicted in Figure 2.3. As can be seen, the packets captured by the observer modules can be directed to filtering and sampling modules implementing different packet selection techniques. One of these modules captures the first packets of each TCP connection, which can be very useful for signature-based worm and botnet detection as well as traffic classification [BMC10]. The flow metering and aggregation modules generate flow records from captured packets or records received by the collector module using configurable sets of flow keys [DM06]. The database writer and reader modules allow storing and retrieving flow records from tables in a database, which is very useful for offline analysis as well as replaying IPFIX traffic. As explained later in Chapters 7 and 10, we use a flow database to generate traffic measurement time series for anomaly detection and to identify the causes of traffic anomalies.

VERMONT is configured with XML configuration files. An innovative feature is the configuration manager which allows to change the configuration at runtime. Dynamic configuration is an important prerequisite for adapting the traffic measurements to the varying needs of the application. As a prototype implementation of the configuration data model proposed in [MCA10], we have implemented a NETCONF agent [EBC$^+$06] which receives configuration requests from a remote management system [MADC06].

TOPAS (Traffic flOw and Packet Analysis System) [BM06, MC07] was originally developed within the European project *DIADEM Firewall* [DIA06]. *TOPAS* implements an IPFIX and PSAMP collector and offers a modular framework for real-time traffic analysis and attack detection. If something interesting is detected, the modules may trigger events to notify the network administrator or to dynamically initiate a reconfiguration of monitoring and analysis functions.

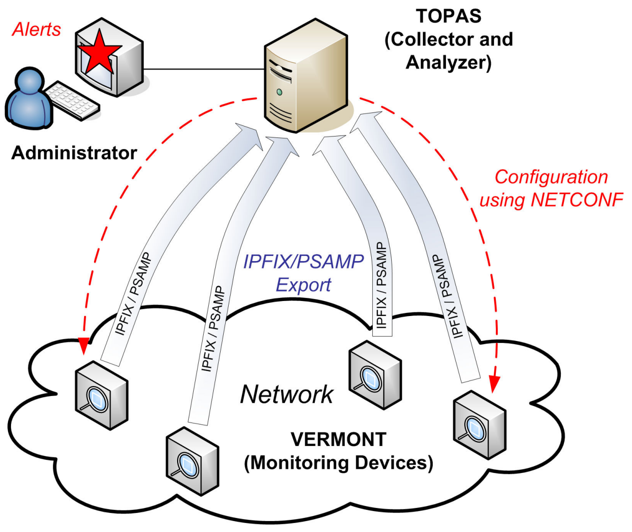


Fig. 2.4: Deployment of *VERMONT* and *TOPAS*

Figure 2.4 shows the deployment of *VERMONT* and *TOPAS* in the network of an ISP. While instances of *VERMONT* act as configurable monitoring devices, *TOPAS* plays the role of the collector and traffic analyzer. In the *DIADEM Firewall* context, this setup was used to detect denial-of-service (DoS) and distributed denial-of-service (DDoS) attacks [MFC⁺05] based on flow records. Later, we used *TOPAS* to analyze header fields and payload of selected packets exported by PSAMP. Therefore, we integrated *Snort IDS* [Roe99] and *Wireshark* [Wir10] into the *TOPAS* framework [MWC07, MC08b].

For offline analysis of flow data, we have developed a web server based system using Java Server Pages (JSP) and Java Servlets. The analyzed data is queried from one or multiple **flow databases** created by *VERMONT*'s database writer. These databases can be located at collectors in different networks as depicted in Figure 2.5. The deployment of distributed flow databases is advantageous in cases where the export of flow data via the IPFIX or *NetFlow* protocol across administrative domains is not possible or where a single database is not fast enough to save the stream of flow records received from the exporters.

2.5 Positioning of this Dissertation

This dissertation focuses on the analysis of flow data with the goal to detect traffic anomalies and malicious traffic. As we only require flow-level measurement data, our work is positioned in the second inner box of Figure 2.2. Flow data can be easily obtained in most networks since many routers and switches implement flow-level measurement functions and sup-

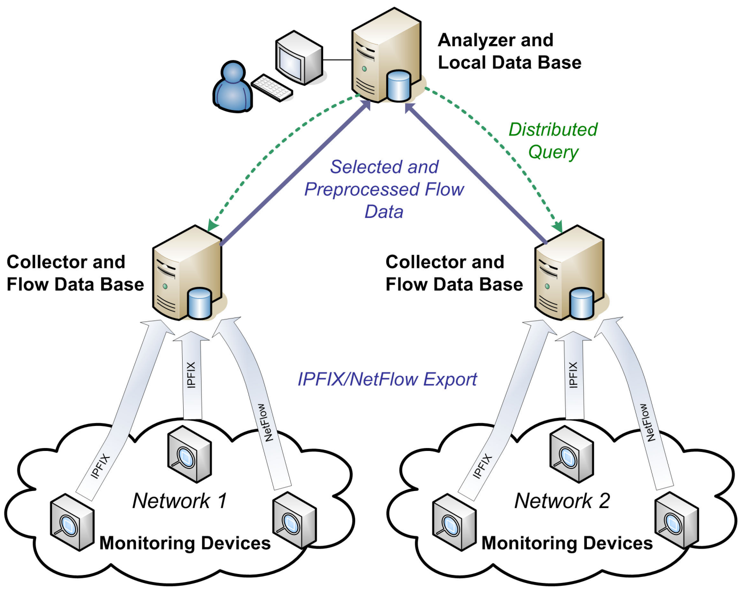


Fig. 2.5: Offline analysis with distributed collectors

port the export of flow records to a collector. Often, flow data is already collected for accounting purposes, which further facilitates the practical deployment of our approach. *VERMONT*, *TOPAS*, and the flow databases provide the necessary infrastructure to implement the considered anomaly detection methods.

Instead of analyzing individual flows, the presented methods detect anomalies in time series of different traffic metrics which are derived from the collected flow records. The analysis of time series has the advantage that the amount of data is independent of the traffic volume and composition. Thus, our approach ensures good scalability and represents a practical solution for anomaly detection in high-speed networks.

Anomaly detection methods often require training data to learn the profile of normal traffic. In contrast, the single-metric anomaly detection methods evaluated in this dissertation do not require any training data as they make use of moving estimators for the mean and standard deviation [MC08a]. Regarding multi-metric anomaly detection, we consider batch-mode PCA, which requires training data to estimate the mean and the correlation matrix, and incremental PCA, which uses moving estimators for this purpose. In all cases, control charts are deployed to detect significant changes in the measurement time series. These control charts are easy to implement and do not require a lot of computational or memory resources.

A general limitation of traffic anomaly detection in time-series is that no information about the causes of the detected anomalies is provided. However, this information can be gathered by inspecting the original flow records.

In Chapter 10, we present three algorithms which search the flow data for typical traffic patterns of common incidents. As a result, the causes of a large proportion of the detected anomalies can be automatically identified without any manual invention of the network administrator.

Part II

FUNDAMENTALS AND RELATED WORK

3. STATISTICAL CHANGE DETECTION

3.1 Introduction

In control engineering, monitoring mechanisms are deployed to observe the properties or behavior of a system and raise an alarm if an important parameter has run out of the range of sound operation. The detection of such events is often associated with the detection of significant changes compared to the normal behavior of the system. The normal behavior can be defined by specification or by measurements of the new and correctly calibrated system. The possible reasons for a change are manifold; examples are failures, malfunctions, and wearout. Independently of the reason, it is often important to quickly react to a detected change, for example by reparing or recalibrating the system, in order to avoid further damage.

In the context of traffic analysis, we are interested in deploying change detection methods for detecting traffic anomalies. The assumption is that a traffic anomaly causes a significant change in certain characteristics of the traffic. However, the utility of the detection results does not only depend on the change detection method deployed. Even more important is the choice of the monitored traffic metrics which must be sensitive to events that are relevant for the operation and administration of the network, such as a network failure, attack or worm traffic. On the other hand, the metrics should be rather insensitive to traffic variation and irregularities caused by legitimate and harmless traffic. Otherwise, we risk to get a large number of irrelevant and uninteresting alarms.

The particularity of change detection is that it considers a series of observations and not individual values. Within this series, change detection searches for a point in time at which a statistical property of the monitored variable changes abruptly. Abrupt means that the change occurs "instantaneously or at least very fast with respect to the sampling period of the measurements" [BN93]. Before and after the change, the statistical properties are assumed to show no or only little variation. Under these conditions, even a small and persistent change can be detected, yet with a longer detection delay than in the case of a large change. This is because more observations need to be collected after the change in order to make a decision with identical confidence.

From a statistical point of view, change detection relies on a **hypothesis test** with the first hypothesis H_0 stating that there is no change, and the second hypothesis H_1 stating the contrary. The design of such a hypothesis test requires a priori knowledge of the probability distribution before the change. Alternatively, the distribution can be estimated from past observations which have to be free of anomalies. In this case, the expected estimation error has to be accounted for, which can have a significant influence on the critical value used as thresholds for test statistics, especially if the estimation is based on a small number of observations only.

A statistical significance test decides if H_0 must be rejected given a sample of one or more observations and a level of significance. The significance level defines the maximum acceptable probability to reject H_0 although the hypothesis is true, which is called a **Type I error** or false alarm. On the other hand, if H_0 is not rejected although the properties have changed, we talk about a **Type II error**. In the case of change detection, an alarm is a false alarm if the statistical properties of the monitored variable have not changed significantly. However, the statistical distinction of false and true alarms may be unrelated to the classification into relevant and irrelevant alarms from the point of view of the analyst.

Change detection methods can be classified according to the following criteria:

Online – offline: Online change detection repeats the tests with the arrival of new observations. The goal is to detect a change with low delay and long time between false alarms. Offline change detection analyzes a series of observations of fixed length. The goal is to decide if there is a change at any point in time. If so, the change time or magnitude may be estimated.

Bayesian – non-bayesian: Bayesian change detection makes use of a priori information about the distribution of the change time in order to improve the detection quality. Non-bayesian change detection does not consider such information, meaning that the probability of a change is assumed to be independent of time.

Parametric – non-parametric: In the parametric case, the probability distribution of the monitored variable is supposed to follow a parametric model. Furthermore, the change is assumed to effect the parameter but not the parametric model itself. On the other hand, non-parametric change detection does not make any assumptions about the distribution of the monitored variable before and after the change. Hence, instead of considering model parameters, non-parametric methods monitor statistic properties of the observations, such as mean, variance, correlation etc., which are assumed to be affected by a change.

Known change – unknown change: If the system model and parameters after change are known a priori, the test procedure can be based on a maximum likelihood decision which determines the sample probabilities for H_0 and H_1 and selects the hypothesis which is more likely. If the magnitude of the change is not known a priori, H_0 can only be rejected with respect to a given 'level of significance'.

Univariate – multivariate: Univariate change detection makes decisions regarding a single variable. Multivariate change detection consider multiple variables and the correlations between them, for example by using the T^2-statistic (see Chapter 5).

In general, the more a priori knowledge is available, the easier it is to detect changes with high accuracy. For example, parametric methods have more power than non-parametric methods, which means that they allow us to detect more true anomalies at the same false alarm level (i.e., probability of an alarm in absence of any significant change). However, if the model assumption is incorrect, parametric methods lose their decisive power and may lead to wrong decisions.

In the case of traffic anomaly detection, we usually do not know the specific distribution of the monitored variables, thus parametric change detection methods must be deployed with care knowing that parametrization tables are usually valid for normally distributed variables only. In general, non-parametric change detection methods are more appropriate. Moreover, changes should be detected very quickly (i.e., online) without requiring any a priori knowledge about their magnitude since such information is usually not available, either. Also, we restrict ourselves to non-bayesian methods and do not assume that changes occur more likely at certain points in time than at others.

In the next sections, we describe different kinds of univariate change detection methods. Multivariate variants of these methods are discussed later in conjunction with the principal component analysis in Chapter 5. Although not fully adequate for our purposes, we start with a quick review of the most important parametric methods in Section 3.2. With help of the log-likelihood ratio, the statistical properties of these methods can be proved independently from any specific distribution family. The theoretical findings represent the basis for the deployment of **control charts** in the area of statistical process control (SPC) [Mon05]. The directions, tables, and rule of thumbs used to set up control charts usually rely on the assumption that observations are independent and normally distributed. A certain robustness against departures from normality is given in control charts of sample means, which is due to the central limit theorem.

Section 3.3 deals with the non-parametric Shewhart, CUSUM (cumulative sum), and EWMA (exponentially weighted moving average) control charts which usually show good detection performance under many kinds of distributions. Furthermore, it sketches how non-parametric tests of goodness-of-fit and non-parametric two-sample tests can be applied to a sliding window of observations. Sections 3.2 and 3.3 contain separate subsections summarizing relevant related work in the area of traffic analysis and anomaly detection.

Section 3.4 concludes this chapter with some preliminary thoughts about the appropriateness of the different change detection methods for traffic anomaly detection.

3.2 Parametric Change Detection Methods

Parametric change detection methods assume that the probability distribution $p_\Theta(x)$ of the observations $\{x_t\}$ is known a priori. This is often the case for engineered systems whose behaviors are well understood and explained by physical laws. In parametric change detection, the change is considered as a change of the distribution parameter (vector) Θ. In the following, the probability before and after the change are $p_{\Theta_0}(x)$ and $p_{\Theta_1}(x)$ respectively. The corresponding parameters Θ_0 and Θ_1 have to be known or must be estimated.

In this section, we focus on parametric control charts. A typical control charts contains a **center line** (CL) representing the average value of the monitored variable under normal conditions. Above and below the center line, the **upper and lower control limit** (UCL, LCL) define the range of normal variation or in-control state. The decision rule detects a change (or out-of-control state) if the observations lie outside this range.

3.2.1 Log-Likelihood Ratio

An essential tool for parametric change detection methods is the logarithm of the likelihood ratio:

$$s(x) = \log \frac{p_{\Theta_1}(x)}{p_{\Theta_0}(x)}$$

Obviously, $s(x)$ is positive if the observation x more likely conforms to the distribution after change $p_{\Theta_1}(x)$ than to the distribution before change $p_{\Theta_0}(x)$, and negative in the opposite case. This can be expressed by the expectation values:

$$E_{\Theta_0}[s(x)] < 0 \text{ and } E_{\Theta_1}[s(x)] > 0$$

Hence, we can define a threshold h for $s(x)$ to decide if the null hypothesis $H_0 : \Theta = \Theta_0$ is accepted or rejected.

For example, if the observations are normally distributed with variance σ^2 and means μ_0 and μ_1 before and after change, $s(x)$ becomes:

$$p_{\Theta_k}(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu_k)^2}{2\sigma^2}}; \quad k \in \{0; 1\}$$

$$\Rightarrow s(x) \quad = \quad \frac{\mu_1 - \mu_0}{\sigma^2}\left(x - \frac{\mu_1 + \mu_0}{2}\right)$$

If $\mu_1 > \mu_0$, $s(x) > h$ is equivalent to the decision rule:

$$x > \mu_0 + L\sigma \quad \text{with } L = \frac{h\sigma}{\mu_1 - \mu_0} + \frac{\mu_1 - \mu_0}{2\sigma}$$

In this equation, the correspondence to the control chart is obvious: μ_0 is the center line and $\mu_0 + L\sigma$ the upper control limit.

As there is usually high variance in a single observation, it is difficult to make an accurate decision with only one sample. Therefore, change detection methods often consider a series of independent observations $\{x_t | a \leq t \leq b\}$ the probability of which can be calculated by $\prod_{t=a}^{b} p_\Theta(x_t)$. With help of the log-likelihood ratio, we obtain the following statistic which is the basis of many control charts and change detection methods:

$$s(x_a, \ldots, x_b) = \sum_{t=a}^{b} s(x_t) = \log \frac{\prod_{t=a}^{b} p_{\Theta_1}(x_t)}{\prod_{t=a}^{b} p_{\Theta_0}(x_t)}$$

$s(x_a, \ldots, x_b)$ is the log-likelihood ratio given the observations $\{x_t | a \leq t \leq b\}$. It is important to note that $s(x_a, \ldots, x_b)$ assumes independent observations. If we expect the x_t to be serially correlated, the statistic provides incorrect results.

Continuing the above example of independent and normally distributed observations, we get:

$$\begin{aligned} s(x_a, \ldots, x_b) \quad &= \quad \frac{\mu_1 - \mu_0}{\sigma^2} \sum_{t=a}^{b} \left(x_t - \frac{\mu_1 + \mu_0}{2}\right) \geq h \\ \overset{\mu_1 > \mu_0}{\Rightarrow} \overline{x} = \frac{1}{N} \sum_{t=a}^{b} x_t \quad &\geq \quad \mu_0 + \left(\frac{\sigma^2 h}{N(\mu_1 - \mu_0)} + \frac{\mu_1 - \mu_0}{2}\right) = \mu_0 + L\frac{\sigma}{\sqrt{N}} \\ &\quad \text{with } N = b - a + 1;\ L = \frac{\sigma/\sqrt{N}}{\mu_1 - \mu_0} h + \frac{\mu_1 - \mu_0}{2\sigma/\sqrt{N}} \end{aligned}$$

This means that a hypothesis test based on $s(x_a, \ldots, x_b)$ corresponds to a control chart of the average value $\overline{x}$. The standard deviation of $\overline{x}$ is $\frac{\sigma}{\sqrt{N}}$. It decreases with increasing N, allowing control limits which are closer to the center line than in the case of individual observations.

3.2.2 Parametric Shewhart Control Chart

The Shewhart control chart of the mean $\overline{x}$ [She31, Mon05] defines upper and lower control limits for averages of N consecutive observations:

$$\overline{x}_l = \frac{1}{N} \sum_{t=(l-1)N+1}^{lN} x_t \quad \text{where} \quad l = 1, 2, \ldots$$

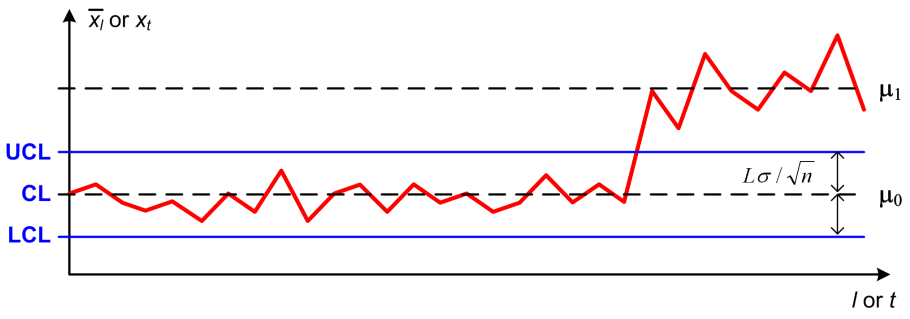


Fig. 3.1: Shewhart control chart

The definition of the center line and the control limits follows from the considerations made in Section 3.2.1. If the observations $\{x_t\}$ are independently and identically distributed with mean μ_0 and variance σ^2 under normal conditions, the mean and variance of $\overline{x}_l$ are μ_0 and σ^2/N. Hence, the control chart defines upper and lower control limits which are typically in the form $LCL = \mu_0 - L\sigma/\sqrt{N}$ and $UCL = \mu_0 + L\sigma/\sqrt{N}$ with tuning parameter L. An alarm is raised if $\overline{x}_l$ passes one of the control limits. Figure 3.1 shows a Shewhart control chart example.

If the distribution of the observations is not known a priori, μ_0 and σ have to be estimated from past observations which must be free of changes and anomalies. Alternatively, moving estimators of μ_0 and σ can be used which are continuously updated with every new observation (cf. Section 4.5). In this case, the control limits are not constant over time, a phenomenon which is called 'adaptive threshold' in some references (e.g., [SP04]) A selection of moving estimators are presented in Section 4.5.

In Section 3.2.1, we have derived the relationship between log-likelihood ratio and decision rule of the upper control limit for independent and normally distributed observations:

$$s(x_{(l-1)N+1}, \ldots, x_{lN}) \geq h \iff \overline{x}_l \geq \mu_0 + L\frac{\sigma}{\sqrt{N}}$$

For a shift of the mean from μ_0 to μ_1, the threshold h corresponding to L can be calculated as $h = \frac{\mu_1-\mu_0}{\sigma/\sqrt{N}}\left(L - \frac{\mu_1-\mu_0}{2\sigma/\sqrt{N}}\right)$. Although this relationship is interesting from a theoretical point of view, we usually do not know μ_1 in practice.

If the observations are independent and normally distributed, $\overline{x}_l$ is normally distributed for any value of N. Hence, we can determine the control limits for a given false alarm rate (or level of significance) α:

$$UCL = \mu_0 + \Phi(1-\alpha/2)\sigma/\sqrt{N}\ ,\ LCL = \mu_0 - \Phi(1-\alpha/2)\sigma/\sqrt{N}$$

Furthermore, we can calculate the in-control and out-of-control **average run length** (ARL), which corresponds to the mean time between false alarms and the mean detection delay. A special case is $N = 1$, the so-called **Shewhart control chart of individuals** which compares individual observations against the control limits.

If the observations follow a non-normal distribution with finite expectation value and finite variance, $\overline{x}_l$ is still approximately normally distributed for large N because the central limit theorem applies. This property is exploited by the non-parametric version of the Shewhart control chart discussed in Section 3.3.1.

For $N > 1$, there exist additional Shewhart control charts for testing variability in the observations, such as the σ-control chart and the range control chart (see [Mon05] for more details). The central limit theorem does not apply in these cases, thus the control limits can only be calculated in the parametric case where the probability distribution is known. Again, a well studied case is the change detection in normally distributed observations.

3.2.3 Parametric CUSUM Control Chart

The parametric CUSUM control chart [Pag54], which is often called CUSUM algorithm, relies on the fact that $S_t = s(x_1, \ldots, x_t)$ has a negative drift under normal conditions and a positive drift after a change. The CUSUM decision rule compares the increase of S_t with respect to its minimum to a threshold h:

$$g_t = S_t - \min_{1 \leq i \leq t} S_i \geq h$$

The same decision can be achieved by testing S_t against the adaptive threshold $(h + \min_{1 \leq i \leq t} S_i)$. The decision function g_t can be rewritten in a recursive form which facilitates the online calculation:

$$g_t = \max\left(0, s(x_t) + g_{t-1}\right) = \left[g_{t-1} + s(x_t)\right]^+ \ , \ g_0 = 0$$

An alarm is raised if g_t exceeds the threshold h. To restart the algorithm, g_t is reset to zero.

From the view of hypothesis testing, the CUSUM control chart repeatedly performs a **sequential probability ratio test** (SPRT) where each decision considers as many consecutive observations as needed to accept one of the hypotheses $H_0 : \Theta = \Theta_0$ and $H_1 : \Theta = \Theta_1$. The CUSUM control chart implicitly starts a new run of SPRT if H_0 has been accepted, and stops with an alarm in the case of H_1. The threshold h allows trading off the mean detection delay and the mean time between false alarms.

As an example, we assume that the observations are independent and normally distributed with variance σ. The mean before change is μ_0, the mean after change μ_1. Then, the decision rule becomes:

$$g_t = \left[g_{t-1} + s(x_t)\right]^+ = \left[g_{t-1} + \frac{\mu_1 - \mu_0}{\sigma^2}\left(x_t - \frac{\mu_1 + \mu_0}{2}\right)\right]^+ \geq h$$

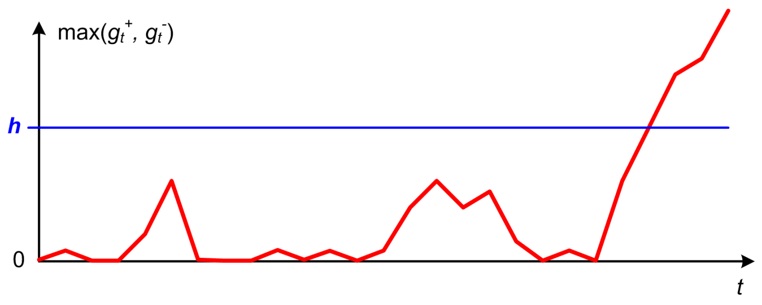


Fig. 3.2: CUSUM control chart

The factor $\frac{\mu_1-\mu_0}{\sigma^2}$ can be omitted by adapting the threshold accordingly:

$$\tilde{g}_t = \left[\tilde{g}_{t-1} + x_t - \frac{\mu_1+\mu_0}{2}\right]^+ = \left[\tilde{g}_{t-1} + (x_t - \mu_0) - \frac{\mu_1-\mu_0}{2}\right]^+ \geq \tilde{h}$$

If μ_1 is not known a priori, $\frac{\mu_1-\mu_0}{2}$ is replaced by reference value (or allowance) K. Typical values are $\sigma/2$ for the reference value and 4σ or 5σ for $\tilde{h}$, resulting in an in-control ARL (i.e., mean time between false alarms) of 168 and 465 respectively [Mon05].

If the observations are independent and identically distributed and if the parameters before and after change Θ_0 and Θ_1 are exactly known, the CUSUM algorithm is asymptotically optimal, meaning that the mean detection delay is minimal if the mean time between false alarms goes to infinity (i.e., for $h \to \infty$) [Lor71]. For correlated observations, we have to replace $p_{\Theta_{0,1}}(x_t)$ by the conditional density $p_{\Theta_{0,1}}(x_t|x_{t-1}, x_{t-2}, ..., x_1)$ when calculating $S_t = s(x_1, \ldots, x_t)$ to maintain the optimality (see [BN93, chap. 8.2]). Otherwise, the probability of false alarms will be increased. It is important to note that CUSUM's optimality is asymptotic and valid only if the necessary conditions are fulfilled. The algorithm is not robust in a sense that it still provides optimal results if the distributions before and after change are not known exactly.

A common property of the Shewhart control charts of individuals and the CUSUM control chart is that both require knowledge of the distribution of observations in order to define precise control limits. Furthermore, independence of observations is assumed (unless the dependencies are exactly known and accounted for). CUSUM is able to detect smaller shifts than the Shewhart control chart because they accumulate over time. However, the shift has to persist for a certain time in order to be detected. Analogously to the upper and lower control limits of the Shewhart control chart, two CUSUM decision functions g_t^+ and g_t^- are needed to detect both, positive and negative shifts of the mean. If the threshold is the same for both functions, the decision rule becomes $\max(g_t^+, g_t^-) \geq h$, which can be illustrated in a single control chart as shown in Figure 3.2.

3.2.4 Further Parametric Algorithms

Without going into details, we mention two additional parametric change detection algorithms used frequently: the generalized likelihood ratio (GLR) algorithm and the Girshick-Rubin-Shiryaev (GRSh) algorithm.

The GLR change detection algorithm [Lor71] calculates the sum of log-likelihood ratios $s(x_a, \ldots, x_b)$ using a maximum likelihood estimate of the parameter after change Θ_1. Hence, the GLR algorithm is asymptotically optimal and can be used if Θ_1 is not known a priori, provided that a minimum magnitude of the changes of interest is given (i.e., a lower limit for $|\Theta_1 - \Theta_0|$).

In contrast to the previously presented algorithms, the parametric GRSh algorithm [GR52, Shi63] relies on the non-logarithmic likelihood ratio $\frac{p_{\Theta_1}(x)}{p_{\Theta_0}(x)}$. The GRSh has very similar optimality properties as the CUSUM algorithm.

3.2.5 Existing Parametric Traffic Anomaly Detection Approaches

Hellerstein et al. apply the GLR algorithm to detect anomalies in the number of webserver requests per five minute interval [HZS98]. Therefore, regular variations are removed from the observations using time-series models to obtain a series of uncorrelated residuals as input to the GLR algorithm. The log-likelihood ratio is determined under the assumption of normally distributed residuals. More details on the deployed time-series modeling are given in Section 4.6.

Thottan and Ji deploy the GLR algorithm for fault detection using different packet and byte counters in the MIB of a router [TJ98]. The counter values are queried every 15 seconds to obtain traffic measurement time series. Serial correlation in windows of 10 measurement values is removed using the AR(1) time-series model. The log-likelihood ratio is calculated for the hypothesis that the residuals of two adjacent windows follow the same multivariate normal distribution. The evaluation of the approach suggests that the proportion of alarms which can be associated with actual network faults is rather small, although additional tests and filters are applied to reduce the number of irrelevant alarms. Section 4.6 provides further information about the AR(1) residual generation applied.

Hajji applies the GLR algorithm in order to detect network faults by monitoring traffic measurement variables, such as packet counts [Haj05]. The distribution of the measurement variables is assumed to follow a finite Gaussian mixture model (GMM). The probability distribution of GMM consists of a weighted superposition of multiple normal subpopulations, which allows approximating many kinds of distribution functions. Hajji proposes an online method for continuously generating new GMM parameter estimates with every new observation. As input to the GLR algorithm, the change (or delta) of the estimates is standardized in order to obtain a baseline variable which exhibits a stationary, zero-mean behavior during normal operation. Due to the limited evaluation of the approach, it is difficult to assess the performance compared to alternative change detection methods.

3.3 Non-Parametric Change Detection Methods

For many of the parametric change detection methods, there exist non-parametric variants that do not assume any specific probability distribution of the observations. Mostly, their properties can only be determined asymptotically, for example for large numbers of observations or for thresholds yielding an infinitesimal false alarm probability. In this section, we present the non-parametric Shewhart control chart, the non-parametric CUSUM control chart, and the EWMA control chart. Finally, we show how common hypothesis tests, particularly two-sample tests, can be used for change detection as well.

3.3.1 Non-Parametric Shewhart Control Chart

As mentioned in Section 3.2.2, the non-parametric Shewhart control chart relies on the central limit theorem which says that the average of a sufficiently large number of independent and identically distributed random variables with finite expectation value and finite variance is approximately normally distributed. So, even if the distribution of the observations is not normal, $\overline{x}_l$ is still approximately normally distributed. Given the mean μ_0 and variance σ^2 of the observations under normal conditions, the upper and lower control limits for $\overline{x}_l$ are the same as in the parametric Shewhart control chart:

$$UCL = \mu_0 + \Phi(1-\alpha/2)\sigma/\sqrt{N} \ , \ LCL = \mu_0 - \Phi(1-\alpha/2)\sigma/\sqrt{N}$$

Of course, these control limits do not hold for small N or if the observations are serially correlated.

3.3.2 Non-Parametric CUSUM Control Chart

The log-likelihood ratio $s(x)$ cannot be calculated in the non-parametric case since $p_\Theta(x)$ is unknown. Therefore, $s(x)$ is replaced by a statistic $u(x)$ with comparable properties: its expectation value of $u(x)$ must be negative under H_0 and positive under H_1. This variant is often called non-parametric CUSUM algorithm. The decision rule becomes:

$$g_t = S_t - \min_{1 \le i \le t} S_i = [g_{t-1} + u(x_t)]^+ \ge h \quad \text{with } S_t = \sum_{i=1}^{t} u(x_i)$$

Again, the threshold h must be adjusted to achieve a satisfactory compromise between low mean detection delay and large mean time between false alarms.

Brodsky and Darkhovsky [BD93] show that the non-parametric CUSUM algorithm is asymptotically optimal if the distribution of $u(x)$ belongs to a family of exponential distributions and if $E_0\left[u(x)\right] = -E_1\left[u(x)\right]$. Under these conditions, the detection delay reaches the theoretic minimum if the mean time between false alarms goes to infinity (which implies $h \to \infty$). However, as precise knowledge about the distribution of $u(x)$ before and after change is needed to draw on this optimality, the parametric CUSUM algorithm could be used just as well. For $E_0\left[u(x)\right] \neq -E_1\left[u(x)\right]$ or a distribution which does not belong to the exponential family, the CUSUM algorithm is not optimal any more. In [BD93], Brodsky and Darkhovsky also prove the asymptotic equivalence of the non-parametric variants of the CUSUM and GRSh algorithms.

In Section 3.2.3, we derived the parametric CUSUM control chart for detecting positive shifts in the mean of normally distributed observations. The same decision rule is commonly deployed in non-parametric CUSUM control charts, which corresponds to the following statistic:

$$u(x) = x - (\mu_0 + K)$$

However, in contrast to the normal case, it is impossible to determine the false alarm probability or the mean detection delay. This means that the properties of the change detection method cannot be derived stochastically. Furthermore, optimality cannot be assumed unless it has been verified that $u(x)$ conforms to one of the above mentioned exponential distributions. Nevertheless, this non-parametric CUSUM control chart can still be deployed as a heuristic change detection method with appropriate parameters found by experimentation. Examples concerning the detection of traffic anomalies are given below in Section 3.3.5.

3.3.3 EWMA Control Chart

The EWMA control chart [Rob59, RH78] relies on the exponential smoothing of observations. Given the smoothing constant λ $(0 < \lambda < 1)$,

$$\tilde{x}_t = \lambda x_t + (1-\lambda)\tilde{x}_{t-1} = \lambda \sum_{i=0}^{t-1} (1-\lambda)^i x_{t-i} + (1-\lambda)^t \tilde{x}_0$$

is a weighted average of all observations up to time t. The weights decrease geometrically with the age of the observation. The starting value is the expected mean before change: $\tilde{x}_0 = \mu_0$. $\tilde{x}_t$ can also be regarded as a moving estimator $\hat{\mu}_t$ of the observations' mean.

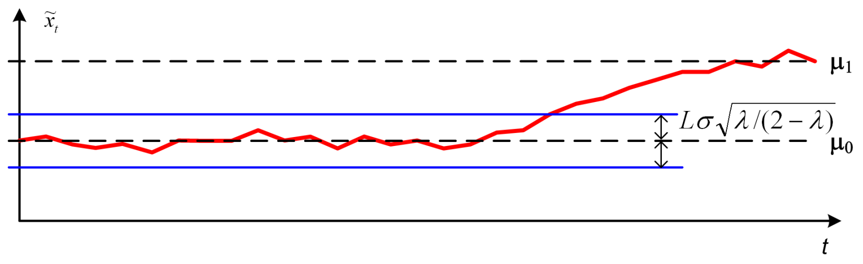


Fig. 3.3: EWMA control chart

If the observations are independent and identically distributed with variance σ^2, the variance of $\tilde{x}_t$ is:

$$Var\left[\tilde{x}_t\right] = \frac{\lambda}{2-\lambda}\left[1-(1-\lambda)^{2t}\right]\sigma^2 \overset{t\to\infty}{\longrightarrow} \frac{\lambda}{2-\lambda}\sigma^2$$

Thus, assumed that the observations have mean μ_0 before change, we can apply the control limits $\mu_0 \pm L\sqrt{Var\left[\tilde{x}_t\right]}$ to $\tilde{x}_t$. This is illustrated in Figure 3.3. If σ is not known a priori, it has to be estimated from training data or with a moving estimator as described in Section 4.5. In Section 4.3.2, we present exponential smoothing as a robust one-step-ahead forecasting method for stationary random processes (i.e., $\hat{x}_{t+1} = \tilde{x}_t$). Hence, the control limits can be interpreted as the maximum allowed deviation of the forecast value from μ_0.

λ and L are design parameters of the control chart. For independent and normally distributed observations, tables and graphs exists which provide the ARL for combinations of λ and L in dependence of the magnitude of the shift in the mean. Popular choices are $2.6 \leq L \leq 3$ and $0.05 < \lambda < 0.25$ (in-control ARL ≈ 500), where smaller λ allow detecting smaller shifts [Mon05].

The EWMA control chart has some interesting properties [Mon05]. It can be tuned to achieve approximately equivalent results as the CUSUM control chart. Secondly, it is quite robust against non-normal distributions of observations, especially for small values of λ (e.g., $\lambda = 0.05$). Finally, after adjusting the control limits, the EWMA control chart still performs well in the presence of low to moderate levels of serial correlation in x_t.

With alternative statistics, similar control charts can be constructed to measure the variability [Mon05]. For example, we can use the EWMS (exponentially weighted mean square error) $s_t^2 = \lambda(x_t - \mu_0)^2 + (1-\lambda)s_{t-1}^2$, which is a moving estimator of the variance ($E[s_t^2] = \sigma^2$ for large t). EWMS is sensitive to both shifts in the mean and shifts in the standard deviation. If we are only interested in changes of the variability, we can replace μ_0 by the EWMA estimate, resulting in the exponentially weighted moving variance (EWMV).

3.3.4 Tests of Goodness-of-Fit and Homogeneity

Control charts are not the only way to detect changes in a series of observations. Usual tests of goodness-of-fit and two-sample tests of homogeneity can be used as well. Tests of goodness-of-fit evaluate if a sample conforms to a given probability distribution while tests of homogeneity check if two samples are likely to follow the same distribution.

In order to apply such tests to a series of observations, the observations have to be grouped into samples. Therefore, we define a sliding window of n_1 recent observations $x_{t-n_1+1}, \ldots, x_t$ which represents the current traffic behavior. This sample is then compared to a given distribution (goodness-of-fit) or another sample of observations (homogeneity) which both are assumed to represent normal traffic behavior. In the two-sample case, we can use a second sliding window of n_2 past observations $x_{t-n_1-n_2+1}, \ldots, x_{t-n_1}$.

The chi-square tests are a family of categorical tests for non-parametric problems. Categorical means that a rather small number of categories exists and that each observation is allocated to exactly one of them. All chi-square tests rely on the calculation of the chi-square statistic Q which is approximately chi-square distributed if the observations in the sample(s) are independent and numerous enough. Designed as a test of goodness-of-fit, the chi-square statistic is calculated as follows:

$$Q = \sum_{i=1}^{m} \frac{(N_i - E_i)^2}{E_i}$$

m is the number of categories, N_i the counted numbers of observations in category i, and E_i the expected number of observations in category i. In a two-sample test of homogeneity, the chi-square statistic becomes:

$$Q = \sum_{j=1}^{2} \sum_{i=1}^{m} \frac{(N_{j,i} - E_{j,i})^2}{E_{j,i}} \quad \text{with } E_{j,i} = \frac{N_{1,i} + N_{2,i}}{n_1 + n_2} n_j \ , \ j \in \{1, 2\}$$

Here, $N_{1,i}$ and $N_{2,i}$ are the numbers of observations in category i in the two samples. $E_{j,i}$ is the expected number of observations in sample j belonging to category i assuming homogeneity of the observations in both samples. In the case of independent and identically distributed observations, Q is approximately chi-square distributed with $(m-1)$ degrees of freedom if $E_{j,i}$ exceeds 5 for all categories and samples [DS01].

The Kolmogorov-Smirnov test is based on a statistic measuring the maximum distance between two cumulative distribution functions (CDF). Again, there exist test variants for goodness-of-fit and homogeneity. In the first case, the empirical CDF of the sample is compared to the CDF of a given distribution. In the second case, the maximum distance between the empirical CDFs of the two samples is determined. With increasing sample sizes, the maximum distance between the CDFs is expected to go to zero.

Moreover, the probability distribution of the distance can be approximated, which allows determining critical values for the Kolmogorov-Smirnov test.

The Wilcoxon-Mann-Whitney ranks test only works as a two-sample test of homogeneity. It sorts all observations to determine their ranks. If identical values occur in the observations (so-called ties), mean ranks are allocated to the concerned observations. If the two samples result from the same distribution, the rank distribution resembles randomly drawn values from $\{1, 2, ..., (n_1 + n_2)\}$. Under this condition, the sum of ranks of the observations S_j of sample j ($j \in \{1, 2\}$) has the following expectation value and variance:

$$E[S_j] = \frac{n_i(n_j + n_2 + 1)}{2} \quad , \quad Var[S_j] = \frac{n_1 n_2(n_1 + n_2 + 1)}{12}$$

When the sample sizes n_1 and n_2 are large enough (≥ 8), the distributions of S_1 and S_2 are approximately normal; if $n_1 + n_2 > 60$, the normal approximation is excellent [MW47, DS01]. The Wilcoxon-Mann-Whitney ranks test rejects the hypothesis of homogeneity if the difference between the calculated rank sum and its expectation value is significant. Considering ranks instead of absolute values, the Wilcoxon-Mann-Whitney test is robust against a small number of outliers in the observations. The discriminating power of the test is large if the samples result from distributions with different means or skewness. On the other hand, the test has low power if the distributions differ in the variance only.

A critical parameter of all tests presented above is the size of the sample(s). Increasing the sample size of recent observations n_1 allows detecting smaller changes if they persist for a long time, but it also increases the detection delay. For two-sample tests of homogeneity, we also need to specify the size of the second sample n_2 which is to be representative for the in-control state. Critical to all these tests is again the necessary condition of independent and identically distributed observations before a change.

3.3.5 Existing Non-Parametric Traffic Anomaly Detection Approaches

The optimality of the CUSUM control chart [BD93] is frequently brought up to justify its usage for traffic anomaly detection. For example, Wang et al. present two approaches to detect TCP SYN flooding attacks. The considered metrics are $\frac{\#SYN - \#FIN}{\overline{\#FIN}}$ [WZS02a] and $\frac{\#SYN - \#SYNACK}{\overline{\#SYNACK}}$ [WZS02b] respectively, where $\#SYN$, $\#FIN$, and $\#SYNACK$ are the numbers of SYN, FIN, and SYN/ACK packets measured in a fixed observation interval, and $\overline{\#FIN}$ and $\overline{\#SYNACK}$ are corresponding moving average values. In the absence of an attack, most SYN packets are followed by a SYN/ACK and a FIN packet, thus the difference between the two counts is small. When comparing SYN and SYN/ACK packets, the counts can be restricted to opposite traffic flows, since the SYN packet and SYN/ACK packet of a TCP handshake are always sent in opposite directions. During a SYN attack, the number of SYN packets significantly exceeds the numbers of FIN packets and SYN/ACK packets, leading to increased metric values. Wang et al. have shown that the approach works fine in different network environments. However, the work lacks evidence suggesting that the condition of independent observations is fulfilled. Since the symmetry between SYN and FIN (or SYN/ACK) packets during normal operation relies on TCP protocol mechanisms, we assume that a visual inspection of the autocorrelation function would show that there is no significant serial correlation in the metric values.

Peng et al. [PLR03] apply the CUSUM algorithm to the number of RST packets returned in response to SYN/ACK packets. This metric yields high values if a reflector attack is going on, caused by SYN packets whose source addresses have been spoofed to the address of the victim. In [PLR04], the same authors count the number of new source IP addresses to detect distributed denial-of-service attacks. Siris and Papagalou [SP04] monitor $\#SYN - \overline{\#SYN}$, where $\#SYN$ and $\overline{\#SYN}$ are the number of SYN packets and the moving average thereof. Although not mentioned in the paper, this corresponds to the prediction error of exponential smoothing (see Sections 4.3.2 and 4.6). They apply the CUSUM algorithm to the monitored metric for detecting SYN flooding attacks. Similarly, Rebahi and Sisalem [RS07] use the number of SIP (Session Initiation Protocol) INVITE messages to detect denial-of-service attacks against SIP servers.

In order to be asymptotically optimal, the CUSUM algorithm must be applied to a time-series of independent observations belonging to a specific family of probability distributions. However, none of the publications shows that these conditions are fulfilled, hence it is unsure if the CUSUM algorithm actually is the most appropriate change detection method for the problems considered.

The research group of Tartakovsky has proposed several approaches to apply the CUSUM control chart to multivariate data, such as packet counts of different packet categories. In [BKRT01] and [TRBK06b], they calculate a chi-square statistic as input for CUSUM in order to detect denial-of-service attacks. For the same purpose, the multichart CUSUM algorithm proposed in [KRT04] and [TRBK06a] performs separate tests on each variable of the multivariate data. Salem et al. apply the multichart CUSUM algorithm to the entries of a count-min sketch to detect SYN flooding attacks and scans [SVG07]. Common to these multivariate methods is the assumption that the variables in the multivariate data are mutually independent, which is usually not fulfilled in the case of traffic measurement data. Tartakovsky at al. also downplay the prerequisite of uncorrelated observations arguing that the false alarm rate decays exponentially fast for increasing thresholds [TRBK06a] under conditions that are to be usually satisfied. Yet, they do not verify if these conditions are fulfilled by the data used in their evaluation.

Ye et al. use EWMA control charts to detect anomalies in computer audit data [YVC03]. The upper and lower control limits are time-invariant and relative to the standard deviation determined out of past observations. The EWMA results are compared to those obtained with a Shewhart control chart of individuals applied to the prediction errors of exponential smoothing[1], a forecasting method which we present in Section 4.3.2. In this case, the control limits are variable and depend on the EWMS estimate of the standard deviation (see Section 4.5). The comparison of the two methods is not very thorough and relies on a single set of testing data for which different parameter settings are tried out. We further discuss the forecasting aspects of this work in Section 4.6. Paul [Pau06] adopts the Shewhart-based approach for detecting denial-of-service attacks against web servers.

The chi-square test of goodness-of-fit has been adopted by Iguchi and Goto [IG99] and by Feinstein et al. [FSBK03] for detecting changes in port activity and packet header field distributions, respectively. Two frequency distributions are compared: a long-term traffic profile and a short-term traffic profile. In contrast to the sliding window approach presented in Section 3.3.4, the frequencies are not calculated from a fixed set of observations but based on exponential smoothing. With each new observation, the short-term frequency distribution N_i ($i = 1, \dots, m$) is updated, applying an exponential decay to all frequencies and increasing the frequency of the new observation's category by one. The long-term profile is calculated similarly but with a much slower decay of old values. The authors of both papers renounce tests against exact critical values because independence of observations and frequency values greater than five cannot be guaranteed.

Krügel et al. [KTK02] calculate the chi-square statistic using the frequency of different characters in packet payloads. Again, the statistic is not compared against any critical values; instead, it is merged into an anomaly score composed of different parts.

Cabrera et al. apply the Kolmogorov-Smirnov test to connection attributes such as the duration and the number of bytes sent in opposite directions [CRM00]. Yet, their analysis goal is not the detection of changes, but to evaluate the discriminating power of the attributes regarding the classification into normal and attack connections.

3.4 Concluding Remarks on Change Detection Methods

An important prerequisite which is common to all the discussed change detection methods is the independence of observations. Furthermore, observations before and after a change are assumed to be identically distributed. Data exhibiting any kind of non-stationarity and serial correlation in the in-

[1] The authors misleadingly call this approach "EWMA control chart for autocorrelated data" although it actually is a Shewhart control chart.

control state do not fulfill these conditions (cf. Section 4.2). If the change detection methods are deployed under such circumstances anyway, the assertions regarding the probability of false alarms do not hold any more.

Internet traffic is affected by seasonal variation and serial correlation. Hence, in order to apply the change detection methods in a reasonable manner, appropriate preprocessing of the measurement data is necessary. Furthermore, suspicious traffic often lasts for a short period of time only, which contradicts the assumption of persistence of the out-of-control state. As a consequence, short occurrences of suspicious traffic risk not to be detected by some of the change detection methods, such as the EWMA control chart.

Non-stationarity which goes back to systematic temporal changes, such as trends and seasonal variation, can be modeled and eliminated using methods from time-series analysis. This approach is explained in the next chapter. If multiple variables are correlated, the application of principal component analysis, which is presented in Chapter 5, can help to separate common temporal changes from the remaining variability in the data. Although neither time-series analysis nor principal component analysis may entirely describe normal traffic behavior, both of these methods allow us to transform the original traffic measurement variables into new variables with much more favorable statistical properties.

The detectability depends on how well the detection method is adapted to a specific change. The monitored statistical property needs to show significantly different values before and after the change. As an example, changes in the variance are difficult to detect if we monitor the sample mean. Similarly, the power of the tests of goodness-of-fit and of homogeneity does not only depend on the magnitude but also on the characteristic of the change.

The most frequently quoted advantage of the non-parametric CUSUM and GRSh algorithms is their asymptotic optimality resulting in minimal mean detection delays at false alarm rates going to zero. However, it is often neglected that this optimality has only been proofed for specific distribution families and under the assumption that the magnitude of the change is known. The optimality property is lost if any of these conditions is not fulfilled.

If the Shewhart control chart considers a subgroup of observations in its decisions, for example by monitoring the mean of N consecutive values, this usually results in longer detection delays compared to CUSUM, GRSh, or EWMA control charts. The same applies to tests of goodness-of-fit and homogeneity since they also make their decisions based on samples of multiple observations. Therefore, these methods are less appropriate for fast change detection.

The complexity and the memory requirements of the different change detection methods differs widely. The Shewhart control chart is very simple as it does not keep any state between subsequent decisions. CUSUM and EWMA control charts keep state information in a single variable (g_{t-1} and

$\tilde{x}_{t-1}$, respectively). The tests of goodness-of-fit and homogeneity operate on the observations in the samples, which requires larger amounts of memory. Also, the test operations are more complex than in the case of control charts. For example, the observations need to be sorted for every run of the Kolmogorov-Smirnov test and the Wilcoxon-Mann-Whitney ranks test.

4. MODELING TEMPORAL BEHAVIOR

4.1 Introduction

As discussed in Chapter 3, most change detection methods require independent and identically distributed observations. Hence, we need means which allow us to estimate if a series of measurements can be assumed to be independent or not. Formally proofing independence with only a limited number of observations is difficult, but we can check if the statistical properties of the measurement data deviate from those of a pure random process. How this can be achieved is described in Section 4.2.

There can be various reasons why measurement data exposes significant deviation from the expected output of a random process. The data may exhibit non-stationarites, which means that the statistical properties measured at different instances in time vary. The conclusion would be that there exist systematic components which lead to observations that are not identically distributed. Apart from non-stationarities, measurement data may show significant serial correlation, indicating dependencies between subsequent values. In order to transform the data into a series of independent and identically distributed observations, models have to be found which explain the systematic components as well as the serial correlation.

In the case of mechanical or electronic systems, such models can be derived from physical laws in conjunction with the internal design of the system and given input parameters. The dynamics of network traffic, however, is influenced by so many factors (e.g., user behavior, protocol mechanisms, queuing and scheduling polices at the inner network nodes, routing decision) that it is very hard to understand and model all relationships and dependencies. In addition, it is not feasible to continuously monitor all the parameters that would be required to estimate and remove their influences on the measurement data. Yet, what we can try is to discover certain temporal regularities in the measurements and describe them by time-series models. As a result, certain variations in the data can be modeled without requiring a deeper understanding of the root causes. In Section 4.3, we give an introduction to time-series analysis and models.

System theory deploys state-space models which explain the dynamic behavior of a system by its internal state vector and an external input vector. It is assumed that the state vector cannot be monitored and modified directly. Hence, keeping the system under control is only possible by imposing appropriate input parameters. Traffic analysis deals with the interpretation and evaluation of measurement data and originally does not aim at actively controlling the behavior of the network. Nevertheless, it is interesting to briefly reflect this alternative modeling approach and to show its relationship to time-series models. As described in Section 4.4, most time-series models can be transformed into linear state-space models and vice versa. Of practical interest for traffic analysis are Kalman filters which have been used to infer traffic characteristics that cannot be measured directly. One such example will be referred to in Section 4.6, together with the presentation of further existing work deploying temporal models for detecting traffic anomalies.

Section 4.7 concludes this chapter by discussing the applicability and usability of time-series analysis for traffic anomaly detection.

4.2 Pure Random Process

The output of a pure random process is a series of independent and identically distributed random variables. This is commonly known as **'white noise'**. It implies that the process is stationary and that there is no serial correlation in the output. The terms **stationarity** and **serial correlation** are explained in the following.

A process X_t is called (strictly) stationary if the joint distribution of $X_{t_1}, ..., X_{t_k}$ is the same as the joint distribution of $X_{t_1+\tau}, ..., X_{t_k+\tau}$ for all $t_1, ..., t_k, \tau$ $(k = 1, 2, \ldots)$. Hence a shift in time has no effect on the distribution. One consequence is that all X_t are identically distributed (case $k = 1$). A stationary process does not exhibit any systematic change in mean or variance, nor any strictly periodic variation. A weaker definition of stationarity, called second-order stationarity, refers only to the first and second-order properties and requires $E[X_{t_1}] = E[X_{t_2}]$ as well as $Cov[X_{t_1}, X_{t_1+\tau}] = Cov[X_{t_2}, X_{t_2+\tau}]$ for all t_1, t_2, τ. In the case of the normal distribution, second-order stationarity implies strict stationarity [Cha03, chap. 3.2.1].

The autocorrelation function is defined as:

$$Corr[X_{t_1}, X_{t_2}] = \frac{E\left[(X_{t_1} - \mu_{t_1})(X_{t_2} - \mu_{t_2})\right]}{\sigma_{t_1}\sigma_{t_2}}$$

where μ_{t_i} and σ_{t_i} are the mean and standard deviation of X_{t_i} $(i = 1, 2)$. If the process is stationary, we have $\mu_{t_1} = \mu_{t_2} = \mu$ and $\sigma_{t_1} = \sigma_{t_2} = \sigma$. In addition, we know that correlation between X_t and $X_{t+\tau}$ is the same for all t and only depends on lag τ. As a consequence, the autocorrelation becomes a function of τ:

$$Corr[X_t, X_{t+\tau}] = \frac{E\left[(X_t - \mu)(X_{t+\tau} - \mu)\right]}{\sigma^2}$$

In the case of a pure random process, X_t and $X_{t+\tau}$ are independent for $\tau \neq 0$. Then, because of $E\left[(X_t - \mu)(X_{t+\tau} - \mu)\right] = E[X_t - \mu]E[X_{t+\tau} - \mu] = 0$, X_t and $X_{t+\tau}$ are uncorrelated as well. Thus, the autocorrelation of a pure random process is:

$$Corr[X_t, X_{t+\tau}] = \begin{cases} 1 & \tau = 0 \\ 0 & \tau \neq 0 \end{cases}$$

In summary, it can be said that the identical distribution of output variables X_t is a necessary condition for stationarity. On the other hand, independence between output variables implies the absence of serial correlation. A random process needs to fulfill the conditions of stationarity and to produce uncorrelated output in order to result in independent and identically distributed observations.

Stationarity as well as the absence of serial correlation can be checked with help of visual chart inspection. Non-stationarities, such as changes in the mean or variance, become obvious when displaying measurement values over time. Furthermore, systematic changes as well as serial correlation have an impact on the **sample autocorrelation**, which is calculated as follows in the time-discrete case:

$$r_\tau = \frac{\sum_{t=1}^{n-\tau}(x_t - \bar{x}_t)(x_{t+\tau} - \bar{x}_t)}{\sum_{t=1}^{n}(x_t - \bar{x}_t)^2}$$

where x_t is the series of observations and n the number of observations. The sample autocorrelation is a biased estimator of the autocorrelation function in the case of stationarity. If r_τ is not decreasing with increasing τ, or if it shows periodic oscillation, the observations do not resemble the output of a stationary random process. In the case of a pure random process, the expectation value and variance of the sample autocorrelation for all $\tau \neq 0$ are $E[r_\tau] \cong -1/N$ and $Var[r_\tau] \cong 1/N$. Values of r_τ lying outside the interval $[-1/N - 2/\sqrt{N}; -1/N + 2/\sqrt{N}]$ are significant at confidence level 95% [Cha03, chap. 4.1]. Hence, if a non-negligible number of r_τ's lie outside this range, the process is not purely random.

4.3 Time-Series Analysis

Time-series analysis looks for statistically significant evidence for trends, seasonal variations, autoregression (AR), or moving average (MA) effects in the measurement data. Therefore, it associates every single measurement to a certain point in time, resulting in a time series. After successfully identifying an appropriate time-series model and estimating the most likely model parameters, the remaining variation in the measurement data - called residual - resembles the output of a pure random process. Under the assumption that the same model which was able to explain the variation in past measurements remains valid for future measurements, time-series models can be used for forecasting upcoming time-series values. In addition, it is possible to estimate the distribution of the prediction error, which is the difference between a measurement and its predicted value. Hence, given the prediction error and a significance level, we can assess if a new measurement fits to the model or not. Just like the residuals, the expected one-step-ahead prediction errors are independent and identically distributed as long as the time-series model is valid. This is why we can use the the prediction errors as input to the change detection mechanisms presented in Chapter 3 to detect significant changes in the traffic behavior.

This introduction to time-series analysis is restricted to the modeling of temporal relationships in time-discrete measurement data. In the following subsection, we present how such temporal relationships can be modeled by setting up a time-series model. Thereafter, Section 4.3.2 focuses on forecasting future values given a specific model.

4.3.1 Time-Series Models

Time-series models approximate the temporal relationships in the measurement data by a combination of systematic changes (trends, seasonal effects), autocorrelation, and innovations caused by a (pure) random process V_t. In order to verify the validity of a time-series model, the **residuals** are determined, which include the part of variation that is not explained by any other component than the innovation. The residuals are given by $(x_t - \hat{x}_t)$, where x_t is the series of measurements and $\hat{x}_t$ is the series obtained by the model when the innovation V_t is set to zero. As can be seen, the residuals form another time series. If the model is valid, the residuals resemble the output of the assumed random innovation process.

The model building process consists of multiple steps:

1. Model and remove systematic changes.
2. Model the remaining serial correlation with an appropriate model of a stationary process.
3. Verify the entire model by checking the residuals.

In order to find a good model, the process is iteratively repeated with varying parameters until the model with the best results is selected at the end. Following the principal of parsimony, models with fewer parameters are preferred if the representation of the data is equally adequate.

Modeling and removing **systematic changes** from the time series is the first step which needs to be performed if the time series shows non-stationary behavior. Non-stationary behavior is reflected by changes in the mean or variance, as well as strictly periodic variations. As an example, a linear trend is modeled as:

$$X_t = a + bt + V_t$$

Additive seasonal variation with period s can be modeled as:

$$X_t = \bar{X} + S_t + V_t \ , \quad S_t = S_{t+s}$$

A universal method for removing trends as well as seasonal variation is differencing:

$$\nabla_s X_t = X_t - X_{t-s} \ , s > 0$$

In order to remove a non-linear trend from the time series, higher-order differencing is necessary:

$$\nabla^n X_t = \nabla^{n-1} X_t - \nabla^{n-1} X_t \ , n > 1 \quad \text{where } \nabla^1 X_t = \nabla X_t = X_t - X_{t-1}$$

After removing all systematic changes, the remaining variation can be described as the output of a stationary process as defined in Section 4.2.

In the second step of the model building process, the remaining **serial correlation** is modeled using a moving average (MA), autoregression (AR), or mixed ARMA process. An important tool to select the right model is the sample autocorrelation. Values of r_τ not coming down to zero reasonably quickly are indicators of non-stationary, which means that further differencing should be applied. If the sample autocorrelation cuts at a certain lag q (i.e., if r_τ is significant only for lags $|\tau| \leq q$) the time series resembles the output of an MA process of order q which is modeled by the following equation:

$$X_t = \sum_{i=0}^{q} b_i V_{t-i}$$

b_i are constants, and V_t is a pure random process with zero mean.

If the sample autocorrelation shows an exponential decrease or resembles an exponentially damped sinusoidal wave, an AR process of order p seems to be appropriate:

$$X_t = \sum_{i=1}^{p} a_i X_{t-i} + V_t$$

a_i are constants. To be second-order stationary, all roots of $z^p - a_1 z^{p-1} - ... - a_p = 0$ must be located in the unit circle, i.e. $|z_i| < 1$. Note that there is a duality between AR and MA processes which allows expressing an AR process by an MA process of infinite order and vice versa.

AR and MA process can be combined into a mixed ARMA process of order (p,q):

$$X_t = \sum_{i=1}^{p} a_i X_{t-i} + \sum_{i=0}^{q} b_i V_{t-1}$$

The autocorrelation function of an ARMA process attenuates without showing a clear cut-off. Again, an ARMA process can be expressed as a pure AR or MA process. However, modeling a time series with an ARMA process often require fewer parameters.

For the sake of completeness, we denote that an ARMA(p,q) process applied to a differentiated time series $\nabla^d X_t$ is called ARIMA process of order (p,d,q). Obviously, an ARIMA process is not stationary in the mean for $d > 0$.

Estimating the process parameters with help of past observations is called 'fitting'. A necessary prerequisite for estimating the properties of a stationary process from a single time series is **ergodicity**, which means that the properties of X_t can be estimated using data collected at other points in time. We will not provide any further details about how to estimate the parameters of specific models and refer to the literature instead (e.g., Chatfield's textbook on time-series analysis [Cha03]).

The verification of the modeling result consists of checking if the residuals $x_t - \hat{x}_t$ are uncorrelated. This can be done by looking for significant sample autocorrelation values r_τ as described in Section 4.2. Alternatively, statistical tests like the t-test and the Durbin-Watson test for AR(1) correlation can be applied (see [Woo03, chap. 12.2] for further details).

4.3.2 Forecasting

As already mentioned, time-series models can be used to forecast future time-series values or to determine confidence intervals of these. Forecasting is based on the assumption that the modeled temporal behavior persists in the future, and that the forecasting uncertainty is solely caused by the innovation process specified in the model. For change detection, we consider the **prediction error** ϵ_t which is the difference between the observation x_t and the corresponding forecast value $\hat{x}_t$ calculated with knowledge about preceding values $\{x_i | i < t\}$. An alert is raised if the prediction error is to high to be explained by the randomness in the model.

With the so-called Box-Jenkins forecasting procedure, future time-series values as well as the expected prediction errors can be foreseen with help of ARIMA models. Given an ARIMA model and the values of past observations and innovations, the minimum mean square one-step-ahead forecast value $\hat{x}_t$ can be directly computed with the model equation by setting the new, unknown innovation V_t to zero. Past innovations are usually unknown, which means that they need to be substituted by the measured prediction errors $\{\epsilon_i | i < t\}$. Because of the difficulties to find the most appropriate ARIMA model, Chatfield discourages from Box-Jenkins forecasting unless the temporal relationship in the measurement data is mainly characterized by short-term correlation [Cha03]. For MA models and series showing trends or seasonal variations, he recommends the usage of simple exponential smoothing, Holt and Holt-Winters forecasting procedures.

Exponential smoothing allows predicting future values by a weighted sum of past observations ($0 < \alpha < 1$):

$$
\begin{aligned}
t = 1 &\quad : \quad \hat{x}_1 = x_0 \\
t > 1 &\quad : \quad \hat{x}_t = (1-\alpha)^{t-1} x_0 + \alpha \sum_{i=1}^{t-1} (1-\alpha)^{t-1-i} x_i = (1-\alpha)\hat{x}_{t-1} + \alpha x_{t-1}
\end{aligned}
$$

This is the same exponentially weighted moving average as used in the EWMA control chart (see Section 3.3.3). The distribution of the weights is geometric and gives more weight to recent observations. Forecasting according to the above equation is optimal for an infinite-order MA (moving average) process $X_t = V_t + \alpha \sum_{i<t} V_i + \mu$, which is equivalent to an ARIMA(0,1,1) process (see [Cha03, chap. 5.2.2]). Yet, exponential smoothing is very robust and also provides good forecasts for other non-seasonal time-series [GJ06]. The optimal value for the smoothing parameter α can be approximated by trying different values and choosing the one with the smallest square sum of the prediction errors.

The **Holt forecasting** procedure combines an exponential smoothing baseline component L_t with a trend component T_t:

$$\hat{x}_t = L_{t-1} + T_{t-1}$$

L_t and T_t are recursively updated according to the following equations:

$$
\begin{aligned}
L_t &= \alpha x_t + (1-\alpha)(L_{t-1} + T_{t-1}) \\
T_t &= \beta(L_t - L_{t-1}) + (1-\beta)T_{t-1}
\end{aligned}
$$

α and β are smoothing parameters which have to be set to appropriate values in the range $(0, 1)$. The **Holt-Winters forecasting** procedure adds an additive or multiplicative seasonal component I_t and a third updating function with smoothing parameter γ. The equations of additive Holt-Winters are as follows:

$$
\begin{aligned}
\hat{x}_t &= L_{t-1} + T_{t-1} + I_{t-1} \\
L_t &= \alpha(x_t - I_{t-s}) + (1-\alpha)(L_{t-1} + T_{t-1}) \\
T_t &= \beta(L_t - L_{t-1}) + (1-\beta)T_{t-1} \\
I_t &= \gamma(x_t - L_t) + (1-\gamma)I_{t-s}
\end{aligned}
$$

The multiplicative Holt-Winters results in:

$$
\begin{aligned}
\hat{x}_t &= (L_{t-1} + T_{t-1})I_{t-1} \\
L_t &= \alpha(x_t / I_{t-s}) + (1-\alpha)(L_{t-1} + T_{t-1}) \\
T_t &= \beta(L_t - L_{t-1}) + (1-\beta)T_{t-1} \\
I_t &= \gamma(x_t / L_t) + (1-\gamma)I_{t-s}
\end{aligned}
$$

For more details, we refer to [Cha03] and the references therein.

As already stated, change detection evaluates the plausibility of the prediction errors. The range of deviations from a forecast value, which can be explained by the randomness of the model, is usually specified relatively to the standard error $se = \hat{\sigma}$ of the residuals observed when fitting the model. If the forecast is unbiased and if the prediction error is assumed to be normally distributed, the so-called prediction interval for confidence level α is given by $\hat{x}_t \pm z_{\alpha/2}\hat{\sigma}$, where $z_{\alpha/2}$ is the $\alpha/2$-quantile of the standard normal distribution.

4.4 State-Space Models

System theory offers an alternative modeling approach based on state-space models. A **linear time-invariant stochastic state-space model** in discrete time is defined by the following transition and observation equations:

$$\begin{aligned} \mathrm{Z}_{t+1} &= \mathbf{F}\mathrm{Z}_t + \mathbf{G}\mathrm{U}_t + \mathrm{W}_t \\ \mathrm{X}_t &= \mathbf{H}\mathrm{Z}_t + \mathbf{J}\mathrm{U}_t + \mathrm{V}_t \end{aligned}$$

Z_t, X_t, and U_t are the state, observation, and input vectors of the system[1], W_t and V_t are two independent error or noise processes whose values are serially uncorrelated. **F** and **H** are the state transition and the observation matrix, **G** and **J** the control matrices. The temporal behavior is defined by the transition matrix **F** and the control matrix **G**. Given the current state (or an estimate for it) and the input, a forecast of the next state can be calculated. Based on this state estimate and the observation matrix **H**, the future observation value can be predicted as well. The state sequence Z_t is said to have the **Markov property** since Z_t depends on Z_{t-1} but not on earlier state values.

Many time-series models can be transformed into linear state-space models. Yet, a necessary condition is that the model composition must be purely additive. In particular, there is an equivalence between state-space models and ARMAX models which are ARMA models extended by an auxiliary input U_t:

$$X_t = \sum_{i=1}^{p} a_i X_{t-i} + \sum_{i=0}^{q} b_i V_{t-1} + \sum_{i=1}^{l} c_i U_{t-i}$$

As an example, a univariate AR(p) process $X_t = \sum_{i=1}^{p} a_i X_{t-i}$ is equivalent to the following state-space process:

$$\begin{aligned} \mathrm{Z}_{t+1} &= \mathbf{F}\mathrm{Z}_t \\ X_t &= \mathbf{H}\mathrm{Z}_t \end{aligned}$$

[1] Usually, the state vector is called X_t, and the observation vector Y_t. We use a different notation in order to maintain consistency with the variables used in other sections.

$$\mathrm{Z}_t = \begin{pmatrix} X_k \\ \vdots \\ X_{t-p+1} \end{pmatrix}, \quad \mathbf{F} = \begin{pmatrix} a_1 & a_2 & ... & a_{p-1} & a_p \\ 1 & 0 & ... & & \\ 0 & 1 & 0 & ... & \\ & & \vdots & & \\ & ... & 0 & 1 & 0 \end{pmatrix}, \quad \mathbf{H} = (1, 0, ..., 0)$$

The equivalence in this simple example is obvious, yet the transformation of a generic ARMA process into a state-space process is more sophisticated. More details can be found in [BN93, chap. 3.2.4].

In the case of engineering problems, **F** and **H** are usually known a priori and deduced from physical laws or models. The observation vector X_t is the measurable output of the system, whereas the value of the state vector Z_t is usually unknown or only partially known as it cannot be observed directly. Hence, the aim is to infer the inner system state from the observable output, which can be achieved with Kalman filters [Cha03, BN93]. System theory also deals with questions about if and how the system can be controlled by setting the right input values.

We can regard traffic measurement data as the observation of a state-space model and the load generated by hosts, applications, services, or users as its input. However, even if we know about these inputs, we have often problems to monitor them. And of course, we are not able to control them. Considering state-space models for traffic analysis can still be useful if the measurement data is the result of hidden inner network states which we actually want to monitor. An example is the traffic matrix. Though, the state-space model requires the observation matrix (i.e., the dependencies between states and observations) and the transition matrix. If these matrices are unknown, they have to be fitted to a given series of observations. The dimensionality of the state vector must be specified before the fitting. After building and fitting the state-space model, a **Kalman filter** can estimate the state vector and track its temporal changes.

4.5 Moving Estimators

Moving estimators are used to estimate the instantaneous statistical properties of a random process. In this section, we summarize how mean, variance, and standard deviation can be estimated using exponential smoothing. We are particularly interested in estimators of the variance and the standard deviation as they can be used to learn about the statistical properties of the prediction errors.

An estimate of the mean is obtained by the **exponentially weighted moving average** (EWMA):

$$\hat{\mu}_t = \rho x_t + (1 - \rho)\hat{\mu}_{t-1} \quad \text{with} \quad 0 < \rho < 1$$

It is identical to the one-step-ahead forecast value of exponential smoothing.

There exist various moving estimators of variance and standard deviation [Mon05]. If the mean μ is known, we can use the **exponentially weighted mean square error** (EWMS) to estimate the variance:

$$\hat{\sigma}_t^2 = \rho(x_t - \mu)^2 + (1-\rho)\hat{\sigma}_{t-1}^2 \quad \text{with} \quad 0 < \rho < 1$$

EWMS is sensitive to an inaccurate value of the mean μ: if μ is not valid, EWMS will be much larger than the actual variance. If μ is unknown or varying over time, we can replace it by the EWMA estimate $\hat{\mu}_t$:

$$\hat{\sigma}_t^2 = \rho(x_t - \hat{\mu}_t)^2 + (1-\rho)\hat{\sigma}_{t-1}^2$$

This variance estimator is called **exponentially weighted moving variance** (EWMV). Using the acronym EWMSV (exponentially weighted moving sample variance), Eyvazian et al. derive a slightly different moving variance estimator with an additional corrective term [ENV07]:

$$\hat{\sigma}_t^2 = \rho(x_t - \hat{\mu}_t)^2 + (1-\rho)\hat{\sigma}_{t-1}^2 + (\hat{\mu}_t - \hat{\mu}_{t-1})^2$$

The effect of the corrective term is that the most recent estimate of the mean $\hat{\mu}_t$ is adopted as mean of all past observations (and not only of x_t).

We can avoid using the quadratic form and use the **mean absolute deviation** (MAD):

$$\Delta_t = \rho|x_t - \mu| + (1-\rho)\Delta_{t-1}$$

For normally distributed observations, the standard deviation can be obtained from the MAD: $\hat{\sigma}_t \approx 1.25\Delta_t$ [Mon05, chap. 9.4].

Given the observations $x_1, \ldots, x_n$, the standard deviation can be estimated from the **average moving range** (normality assumed) [Mon05, chap. 5.4]:

$$\overline{MR} = \frac{1}{(n-1)} \sum_{t=2}^{n} |x_t - x_{t-1}| \quad , \quad \hat{\sigma} = 0.8865\overline{MR}$$

Similarly, the variance can be estimated from the **mean square successive difference** (MSSD):

$$\hat{\sigma}^2 = \frac{1}{2(n-1)} \sum_{t=2}^{n} (x_t - x_{t-1})^2$$

Accordingly, we can construct moving estimators which are based on the difference of successive observations:

$$\overline{MR}_t = \rho|x_t - x_{t-1}| + (1-\rho)\overline{MR}_{t-1} \quad ; \quad \hat{\sigma}_t = 0.8865\overline{MR}_t$$

$$\hat{\sigma}_t^2 = \rho\frac{(x_t - x_{t-1})^2}{2} + (1-\rho)\hat{\sigma}_{t-1}^2$$

Range and difference-based estimators are robust against inaccuracies in the estimated mean but sensitive to serial correlation in the observations.

4.6 Existing Traffic Anomaly Detection Approaches Using Time-Series Analysis

Hood and Ji [HJ97] convert MIB variables into a measurement time series and eliminate serial correlation by fitting an AR(2) model on a sliding window of 20 observations. The estimated parameters represent a new multivariate time series of features which is then used for detecting network failures with help of Bayesian networks.

Thottan and Ji fit an AR(1) model for non-overlapping windows of 10 observations [TJ98]. The choice of AR(1) is justified by the small number of observations which does not allow estimating more complex time-series models. The residuals are assumed to follow a multivariate normal distribution. As mentioned in Section 3.2.5, the GLR algorithm is used to decide whether the distribution of the residuals in two adjacent windows is identical or different. While the focus of Thottan and Ji is on network fault detection, Wu and Shao adopt the same methodology for traffic anomaly detection [WS05].

An exemplary application of time-series modeling is given by Hellerstein et al. in [HZS98]. Daily and weekly seasonality as well as monthly trend in webserver requests measured per five minute interval are removed before an AR(2) model is fitted to eliminate the remaining serial correlation. In contrast to Holt-Winters, seasonal and trend components of the model are not recursively updated using exponential smoothing, but estimated once from a set of training data using ANOVA (analysis of variance) and linear regression. It is interesting that 64 percent of the variability in the training data are covered by seasonality and trend, which confirms the common observation that modeling these two components is very effective. The GLR change detection algorithm is applied to the resulting residuals as described in Section 3.2.5.

Brutlag [Bru00] employs Holt-Winters forecasting to model baseline, trend, and daily variation in the outgoing traffic of a web server. The expected deviation of the forecast value from the observation is continuously estimated by MAD (see Section 4.5) using the same smoothing parameter as for updating the seasonal component. An alarm is raised if the measured value lies outside the prediction interval for a certain number of times within a moving window of fixed length. Barford et al. [BKPR02] apply Holt-Winters forecasting to time-series of packet, byte, and flow counts. The authors' actual goal is to evaluate the performance of another anomaly detection approach which is based on wavelet analysis and deviation of local variance (deviation score). The evaluation yields similar detection performance for the two approaches.

As mentioned in Section 3.3.5, Ye et al. use EWMA and Shewhart control charts to detect anomalies in computer audit data [YVC03]. The control charts are not applied to individual observations but to their EWMA estimates in order to reduce the variability. In addition, the authors deploy exponential smoothing to produce a series of prediction errors as input to the Shewhart control chart. The Shewhart control limits are relative to the prediction error's standard deviation estimated by EWMS. The interval between the control limits can be regarded as the prediction interval of the forecast value, thus the change detection relies on the detection of inaccurate forecasting results.

In Section 3.3.5, we also mentioned the work of Siris and Papagalou who use EWMA as a moving estimator of the mean number of SYN packets per time interval. Change detection methods are applied to the difference of the EWMA and the measured value, which is the one-step-ahead prediction error of exponential smoothing.

Soule et al. [SST05] use a Kalman filter for estimating the traffic matrix from SNMP link measurements given the routing information. Anomalies are detected based on the Kalman prediction error using different methods like GLR change detection algorithm, wavelet decomposition, and deviation score (see Barford et al. [BKPR02]).

4.7 Concluding Remarks on Time-Series Analysis

We motivated the application of temporal modeling approaches with the prerequisite of independent observations required by most change detection methods. Most kinds of traffic measurement data do not fulfill this condition, thus change detection methods cannot be applied directly. Hence, the idea is to find a model which explains the temporal dependencies in the measurement data. By removing the modeled effects, we hope to yield a new time series which better conforms to the constraints of the detection methods. The models are typically used to forecast future observations and to assess the conformance of new measurements based on the prediction error.

The main problem with this approach is that the model shall describe normal traffic behavior. In order to fit such a model, we need measurement data which is free of anomalies and changes. In practice, it is very difficult to obtain anomaly-free data covering longer periods of time. Even if we want to eliminate the prevalent effect of daily or weekly variation only, the estimation of the corresponding model parameters is not trivial because anomalies occur nearly every day or week. Therefore, robust techniques which continuously learn and update the model parameters, such as exponential smoothing, Holt, and Holt-Winters, are the most appropriate. This explains why most publications proposing forecasting for traffic anomaly detection rely on these techniques. Exponential smoothing, Holt, and Holt-Winters also depend on some parameters which have to be set appropriately (namely the smoothing parameters), yet the optimal setting is much less critical than in the case of ARIMA models.

Despite of the potentials of modeling the temporal dynamics of traffic measurements, there is one conceptional limitation: time-series analysis tries to explain non-stationarity by trend or seasonal effects. However, there exist other events causing systematic changes as well, for example, patch days of widely deployed software and operating systems. If we knew about these events, when they occurred, and how they influenced the traffic, we could model them as input of a state-space model or as the explanatory variables of a multiple linear regression model. Although it is likely that we know the time of certain events, getting reliable estimates of their effect on the traffic is difficult.

Because of these difficulties, modeling systematic changes beyond trend and seasonality is out of scope of our work. As a consequence, we have to be aware of the fact that the considered models do not reflect the entirety of variation in normal traffic. Hence, the resulting residuals are not always stationary under normal conditions, which may lead to false alerts in the change detection process.

Chapter 6 discusses long-range dependence which is observed in certain Internet traffic characteristics. As we will see, the presence long-range dependence may be advantageous for time-series forecasting and anomaly detection.

5. MODELING CORRELATION WITH PRINCIPAL COMPONENT ANALYSIS

5.1 Introduction

Traffic measurements often result in multivariate observations where the different variables measure the same traffic metric at different locations in the network, multiple traffic metrics at one location in the network, or even multiple traffic metrics at different locations in the network. The monitored variables are usually correlated because the different metrics as well as the traffic measured at different locations are not independent. As an example, if the same traffic flow is monitored by multiple network monitors, the measurements will be correlated. An example for correlation among different metrics is the number of packets and the number of bytes observed in a given time interval. Traffic anomaly detection can take into account the relationships among different variables in order to detect changes in the correlation structure.

Principal component analysis (PCA) is a technique to analyze the correlation structure among different variables. It performs an affine transformation of the original coordinate axes onto the **principal components** (PCs). With this transformation, the original variables are transformed into a new set of uncorrelated variables, called PC **scores**. The PCs are ordered according to decreasing proportion of variability they contribute to the original variables. The first PCs contributing the largest part of the variability describe the main correlation relationships in the data. The score time series of these PCs can be interpreted as the output of latent variables that have a strong influence on one or more original variables. PCA is typically used to identify this set of PCs as a model for the correlation structure in the data. The remaining PCs cover the residual variability which is often considered as noise.

The PCs are usually determined from the **covariance matrix** of the original variables. As this matrix is typically not known a priori, it has to be estimated from observations. In the case of 'batch-mode PCA', the sample covariance matrix is calculated using a set of historic observations called 'training data'. The amount of observations in the training data has to be sufficiently large to get estimates with appropriate accuracy. An alternative solution is to use moving estimators, which leads to a PCA variant called 'incremental PCA' in the literature. Here, the covariances and the PCs are continuously re-estimated with every new observation.

Statistical tests allow us to verify if a single observation or a group of multiple observations conforms to the correlation structure modeled from the training data. With these tests, we can detect outliers as well as observations that deviate in the main directions of correlation. As we consider time series of observations, these test can be sequentially performed on every new observation using appropriate control charts, for example the T^2 control chart (see later in Section 5.2.3).

Systematic changes which affect many of the original variables are typically covered by a much smaller number of PCs while the time series of the other PC scores do not exhibit these changes. Thus, PCA allows detecting seasonal variation in the data and separating it from the remaining variability. Existing traffic anomaly detection approaches have exploited this property.

Section 5.2 provides an introduction to PCA explaining the underlying transformation as well as the possible applications for outlier detection, dimensionality reduction, and residual analysis. In Section 5.3, we address particular aspects, such as how to deal with variables measured at very different scales, possible ways to estimate the covariance matrix, and the effect of systematic changes on the PC scores. A survey of existing approaches deploying PCA for traffic anomaly detection is given by Section 5.4. Section 5.5 finally discusses the requirements, problems, and limitations of PCA.

5.2 Fundamentals of PCA

PCA relies on an affine transformation of multivariate data into a set of uncorrelated variables called scores. In Section 5.2.1, we explain how to find the appropriate transformation matrix. One reason for the transformation is the deployment of multivariate control charts, such as the control chart of Hotelling's T^2 statistic described in Section 5.2.3. Another application of PCA is to model the correlation relationship between the original variables with a small subset of the PCs. Deviations from the model can then be detected with the Q statistic and Hawkins' T_H^2 statistic as explained in Section 5.2.4. For additional information about PCA and its applications, we refer to Jackson's user's guide to principal components [Jac91].

5.2.1 Principal Component Transformation

Geometrically, the principal component transformation is a translation which moves the data mean into the point of origin, followed by a rotation which turns the principal components of the data onto the coordinate axes. Finally, the scaling of the new coordinates may change. The principal components are ordered by decreasing proportion of explained variability.

In order to find the transformation matrix, we need to calculate the eigenvalues l_i and eigenvectors t_i of the covariance matrix $\mathbf{\Sigma}$ of the original variables. Given the number of variables p, the covariance matrix is a symmetric $p \times p$ matrix. As the covariance matrix is usually not known, it is estimated by the **sample covariance matrix S** using a given set of observation vectors $\{\mathrm{x}_t | t = 1, \dots, n\}$ which we call **training data**. This approach is called **batch-mode PCA**; an alternative solution using moving estimators will be discussed later in Section 5.3.3. The elements of $\mathbf{S}$ are calculated as follows $(i, j = 1, \dots, p)$:

$$s_{i,j} = \frac{\sum_{t=1}^{n}(x_{t,i} - \overline{x}_i)(x_{t,j} - \overline{x}_j)}{n-1} = \frac{n\sum_{t=1}^{n} x_{t,i}x_{t,j} - \sum_{t=1}^{n} x_{t,i}\sum_{t=1}^{n} x_{t,j}}{n(n-1)}$$

$x_{t,i}$ is the observation of the i-th variable at time t, $\overline{x}_i$ the mean of all observations of the i-th variable. A non-zero sample covariance $s_{i,j}$ indicates that the i-th and j-th variable are correlated. The diagonal elements $s_{i,i} = \hat{\sigma}_i^2$ are the estimated variances of the original variables.

The eigenvalues l_i and eigenvectors t_i are determined by solving the following equations:

$$|\mathbf{S} - l_i\mathbf{I}| = 0 \quad ; \quad (\mathbf{S} - l_i\mathbf{I})\,\mathrm{t}_i = 0$$

where $\mathbf{I}$ is the identity matrix. The eigenvectors t_i represent the PCs in the coordinate system of the original variables. They are orthogonal to each other. Their elements are also called **loadings**.

The covariance matrix $\mathbf{\Sigma}$ is always positive semi-definite, which means that one eigenvalue is positive, the remaining eigenvalues are positive or zero. However, the covariance matrix also needs to be nonsingular (invertible) in order to apply PCA, which means that no variable may be a linear combination of the others. If this condition is fulfilled, the covariance matrix is positive definite, that is, there are p positive eigenvalues which can be ordered decreasingly: $l_1 \geq l_2 \geq \ldots \geq l_p > 0$. Working with sample covariance matrices, zero eigenvalues are improbable even in existence of linear relationships. Thus, the existence of very small positive eigenvalues may already suggest that the covariance matrix is singular and that PCA should not be used.

The determinant of the sample covariance matrix corresponds to the product of its eigenvalues: $|\mathbf{S}| = \prod_{i=1}^{p} l_i$. An interesting property is that the sum of eigenvalues equals the sum of variances of the original variables: $\sum_{i=1}^{p} \hat{\sigma}_i^2 = \sum_{i=1}^{p} l_i$. The ratio $l_i / \sum_{i=1}^{p} l_i$ reflects the proportion of variability contributed by the i-th PC. Hence, the eigenvector of the largest eigenvalue t_1 approximates the direction of largest variability (also called 'line of best fit') in the original data, the eigenvectors of the second largest eigenvalue t_2 the direction of second largest variability, and so on.

PCA is efficient if a small subset of the of PCs covers most of the variability, which is the case if the original variables are highly correlated. To assess in advance if PCA is worthwhile, we can calculate lower and upper boundaries of the largest eigenvalue l_1 with help of the sample covariance matrix **S**:

$$\max_i(\sigma_i^2) \leq l_1 \leq \max_i \sum_{j=1}^{p} |s_{i,j}|$$

Hence, if the given range indicates that the first PC contains a large proportion of the overall variability, the application of PCA is promising.

There exist various scalings of the eigenvectors. Normalizing the eigenvectors to unit length ($\mathrm{u}_i = \frac{\mathrm{t}_i}{|\mathrm{t}_i|}$), we obtain an orthonormal matrix $\mathbf{U} = (\mathrm{u}_1 \, \dots \, \mathrm{u}_p)$ which transforms the original variables into **z-scores** [Jac91]:

$$\mathrm{z} = \mathbf{U}'(\mathrm{x} - \bar{\mathrm{x}})$$

In the above equation, x is the observation vector, $\bar{\mathrm{x}}$ the vector of means $\overline{x}_i$, and z the vector of the z-scores. The subtraction of the means translates the centroid of the original variables to the origin of the coordinate system. As a consequence, the z-scores have zero mean whereas the variance of the i-th component z_i is l_i. The inverse transformation converts the z-scores into the original variables:

$$\mathrm{x} = \mathbf{U}\mathrm{z} + \bar{\mathrm{x}}$$

If the lengths of the eigenvectors are scaled to $1/\sqrt{l_i}$, we obtain a matrix of w-vectors which transforms the original variables into **y-scores**:

$$\mathrm{y} = \mathbf{W}'(\mathrm{x} - \overline{x}) \quad \text{with } \mathbf{W} = (\mathrm{w}_1 \, \dots \, \mathrm{w}_p)\,, \quad \mathrm{w}_i = \frac{1}{\sqrt{l_i}}\mathrm{u}_i$$

y-scores have unit variance and facilitate the calculation of the T^2 statistic presented in the next section. As an interesting property, the inverse covariance matrix can be determined as $\mathbf{S}^{-1} = \mathbf{W}\mathbf{W}'$ (i.e., the inverse can be calculated from the eigenvectors of **S**).

As a third option, the eigenvectors can be scaled to lengths $\sqrt{l_i}$ resulting in v-vectors with coefficients in the same units as the original variables:

$$\mathbf{V} = (\mathrm{v}_1 \, \dots \, \mathrm{v}_p)\,, \quad \mathrm{v}_i = \sqrt{l_i}\mathrm{u}_i$$

This can be useful to interpret the correlation relationships or for data modeling. Moreover, the inverse transformation from y-scores to the original variables is based on **V**:

$$\mathrm{x} = \mathbf{V}\mathrm{y} + \bar{\mathrm{x}}$$

5.2.2 Tests on Individual PC Scores

The PC scores represent a set of independent variables resulting from different linear combinations of the original variables. Tests and control charts can be applied to the scores of an individual PC, yet the definition of exact critical values and control limits requires knowledge about the score's distribution. Jolliffe writes in his textbook on PCA that the distribution of the PC scores can be assumed to be approximately normal due to the central limit theorem for large p [Jol02, chap. 10.1, p. 236]. However, the validity of this assumptions is evident only for identically distributed original variables with weak correlation. Jolliffe does not deliver any proof for the general case.

In the special case of multivariate normally distributed original variables, we know for sure that the PC scores are normally distributed as well. If we apply a tests to the different PC scores i with significance levels α_i, the overall probability for Type I error is $1 - \prod_i (1 - \alpha_i)$ because of the independence of the PCs [Jac91, chap. 1.7.2].

5.2.3 Hotelling's T^2-Test

Hotelling's T^2-test [Hot31] is a multivariate generalization of the Student t-test. The null hypothesis assumes that an individual multivariate observation conforms to a given mean and standard deviation. The **T^2 statistic** is the squared norm of the y-scores, yet it can also be calculated from z-scores or the original measures:

$$T^2 = \mathrm{y}'\mathrm{y} = \sum_{i=1}^{p} \frac{z_i^2}{l_i} = (\mathrm{x} - \bar{\mathrm{x}})' \mathbf{S}^{-1} (\mathrm{x} - \bar{\mathrm{x}})$$

If x is multivariate normal, critical values of T^2 can be derived from the F-distribution:

$$T^2_{p,n,\alpha} = \frac{p(n-1)}{n-p} F_{p,n-p,\alpha}$$

where n is the number of observations used for estimating $\mathbf{S}$ and $\bar{\mathrm{x}}$. α is the level of significance. The limits of T^2 compose an ellipsoid in original coordinates, and a sphere in y-scores.

According to Montgomery [Mon05], the equation above gives a good approximation of the critical values for $n > 100$. For smaller n, the exact formula $\frac{p(n+1)(n-1)}{n(n-p)} F_{p,n-p,\alpha}$ should be used. As $\lim_{n\to\infty} F_{p,n-p,\alpha} = \frac{1}{p}\chi^2_{\alpha,p}$, the χ^2-distribution approximates the distribution of T^2 quite well for very large n. If the sample covariance matrix $\mathbf{S}$ and the sample mean $\overline{\mathrm{x}}$ are replaced by the population's covariance matrix and the expectation values, T^2 is exactly χ^2-distributed with critical values $\chi^2_{\alpha,p}$.

The reason for an observation being out of the T^2 limits can be inspected by identifying the responsible PCs, and in a further step the original variables that are the most correlated with it. Runger et al. [RAM96, Mon05] propose to determine the relative contribution of the i-th original variable directly using the following indicator:

$$d_i = T^2 - T^2_{(i)}$$

where $T^2_{(i)}$ is the T^2 statistic of all variables except the i-th one. Variables with the largest values d_i contribute for the most part to T^2.

If the original variables are not multivariate normal, the T^2-test can be applied to subgrouped data (called "group data" in [Jac91]). In this case, the T^2 statistic is calculated from the distance of the subgroup mean $\overline{\mathrm{x}}$ to the known or estimated mean of the population $\overline{\overline{\mathrm{x}}}$:

$$T^2 = m \left(\overline{\mathrm{x}} - \overline{\overline{\mathrm{x}}}\right)' \mathbf{S}^{-1} \left(\overline{\mathrm{x}} - \overline{\overline{\mathrm{x}}}\right)$$

where m is the number of observations in the subgroup. If the original variables have finite expectation values, variances, and covariances, the subgroup mean is approximately multivariate normal for sufficiently large m due to the central limit theorem. If the normality assumption of the subgroup mean holds and if the population's covariance matrix and expectation values are used instead of their estimates, T^2 is χ^2-distributed. If the covariance matrix and the mean are estimated from training data (which may be subgrouped as well), the distribution of T^2 can be derived from the F-distribution again, but with a different degree of freedom as in the T^2-test of individual observations. For more details, we refer to Montgomery [Mon05, chap. 10.3] and Jackson [Jac91, chap. 6].

The T^2 statistic can be deployed in a so-called **Hotelling T^2 control chart** for detecting shifts in the mean and variance [Mon05]. In this case, T^2 is the monitored variable in the control chart. Like the Shewhart control chart (see Section 3.2.2), we can distinguish the T^2 control chart of individual observations and the T^2 control chart of observation subgroups. The upper control limit of the T^2 control chart corresponds to the critical value of the corresponding T^2-test. A lower control limit does not exist.

The T^2 statistic can also be deployed in a **multivariate EWMA control chart** (MEWMA control chart). The MEWMA control chart as proposed by Lowry et al. [LWCR92] corresponds to the T^2 control chart with $(\mathrm{x}_t - \overline{\mathrm{x}})$ replaced by an exponentially weighted moving average:

$$T_t^2 = \tilde{\mathrm{x}}_t' \tilde{\mathbf{S}}^{-1} \tilde{\mathrm{x}}_t \quad \text{with } \tilde{\mathrm{x}}_t = \lambda(\mathrm{x}_t - \overline{\mathrm{x}}) + (1-\lambda)\tilde{\mathrm{x}}_{t-1}$$

The covariance matrix of $\tilde{\mathrm{x}}_t$ is:

$$\tilde{\mathbf{S}} = \frac{\lambda}{2-\lambda}\left[1-(1-\lambda)^{2t}\right]\mathbf{S} \quad \overset{t\to\infty}{\longrightarrow} \quad \frac{\lambda}{2-\lambda}\mathbf{S}$$

For a constant size of shift and a constant in-control ARL (average run length), the out-of-control ARL increases with an increasing number of variables p, which means that changes are detected with increasing delay. This problem can be mitigated by reducing the dimensionality and considering only a subset of the PCs in the T^2 statistic as described in Section 5.2.4.

Several proposals for **multivariate CUSUM control charts** exist which make use of the covariance matrix to account for correlation relationships in the data [Cro88, PR90]. However, the proximate solution to build a CUSUM statistic of T^2 values does not lead to the best change detection performance [Cro88]. Since the multivariate CUSUM control charts have similar properties as the MEWMA control, we do not go into more details here.

It is important to note that the same limitations as for univariate control charts apply (cf. Section 3.4). In particular, the multivariate time series must be free of systematic changes and serial correlation. If these conditions are not fulfilled, the formulas for calculating the critical values are invalid. We discuss possible utilizations of PCA in the presence of systematic changes in Section 5.3.4.

5.2.4 Reducing Dimensionality and Residual Analysis

The orientation of the PCs belonging to identical eigenvalues is undefined. The corresponding PCs span a subspace in which the data does not show any correlation relationship. In practice, eigenvalues calculated for a sample covariance matrix are very unlikely to be exactly the same, yet some of them may be approximately equal. Typically, this concerns the smallest eigenvalues while the first eigenvalues are well separated.

The idea of reducing the dimensionality by retaining only PCs of the first well separated eigenvalues relies on the assumption that the remaining variability is caused by an uncorrelated error process. If we retain the first $k < p$ PCs, the resulting transformations into y-scores and back again are as follows:

$$\mathrm{y} = \mathbf{W}'(\mathrm{x} - \bar{\mathrm{x}}) \quad \text{with } \mathbf{W} = (\mathrm{w}_1 \dots \mathrm{w}_k)$$

$$\hat{\mathrm{x}} = \mathbf{V}\mathrm{y} + \bar{\mathrm{x}} \quad \text{with } \mathbf{V} = (\mathrm{v}_1 \dots \mathrm{v}_k)$$

Now, y is a vector of length k and $\hat{\mathrm{x}}$ an estimate of x which contains the part of variability covered by the retained PCs. Among all sets of k PCs, the k first PCs have maximum correlation with the original variables and maximize the explained variability.

The T^2-test can be applied to the subset of retained PCs using $T^2 = y'y$ to detect outliers in the k first PCs. Assuming multivariate normality, $T^2_{k,n,\alpha}$ provides the corresponding critical value. T^2 and MEWMA control charts can be designed in an analogous manner using only the retained PCs. However, anomalies detected in the first PCs are typically also visible in the original variables as they affect the main directions of variability in the data. Furthermore, if such outliers appear in the training data, they very likely affect the result of PCA as they inflate variances and covariances. In

Section 5.3.2, we discuss different ways how the influence of these anomalies on the modeled correlation structure can be reduced.

Residual analysis examines the deviation of the observation from the variability modeled by the retained PCs. The difference between x and $\hat{\mathrm{x}}$ constitutes the residual vector ϵ:

$$\epsilon = \mathrm{x} - \hat{\mathrm{x}} = \mathrm{x} - (\mathbf{V}\mathrm{y} + \bar{\mathrm{x}}) = (\mathrm{x} - \bar{\mathrm{x}}) - \mathbf{V}\mathbf{W}'(\mathrm{x} - \bar{\mathrm{x}})$$

The residual vector can be used to check if the given observation x is adequately characterized by the PCs in the model. For this purpose, two test statistics have been proposed: the **Q statistic** and **Hawkins' T_H^2 statistic** [Haw74]:

$$Q = |\epsilon|^2 = (\mathrm{x} - \hat{\mathrm{x}})'(\mathrm{x} - \hat{\mathrm{x}}) = \sum_{i=k+1}^{p} l_i y_i^2 = \sum_{i=k+1}^{p} z_i^2$$

$$T_H^2 = \sum_{i=k+1}^{p} y_i^2 = (\mathrm{x} - \bar{\mathrm{x}})'\mathbf{S}^{-1}(\mathrm{x} - \bar{\mathrm{x}}) - \sum_{i=1}^{k} y_i^2$$

If the original data x is multivariate normal, the approximate distribution of Q can be derived from the normal distribution while T_H^2 has a T^2-distribution (see [Jac91, chap. 2.7] for more details). According to Jackson [Jac91], the two tests are more or less equivalent.

Anomalies detected in the residuals (or last PCs) typically represent individual outliers which are not apparent in the original variables since the corresponding PCs contribute very little variability. Of course, such small outliers in the training data disturb the calculation of the PCs if they appear frequently. As a result, the affected PCs may no longer be among the last PCs but catch up with the first PCs. Hence again, it is necessary to dispose of training data that is mostly free of anomalies or to reduce their influence on the estimated covariance matrix (cf. Section 5.3.2).

What needs to be decided is the number of PCs retained in the model. A variety of so-called **'stopping rules'** has been proposed that indicate when to stop retaining PCs starting from the first one. For the sake of brevity, we confine ourselves to one popular graphical method by Cattell called scree test [Cat66]. The **scree test** plots the ordered eigenvalues l_i next to each other at an absolute or logarithmic scale. If a break can be observed for a specific eigenvalue l_k such that the remaining eigenvalues $l_{k+1}, \ldots, l_p$ decline very slowly, Cattell proposes to retain the PCs of the eigenvalues $l_1, \ldots, l_k$. A description of alternative methods for determining the number of PCs can be found in [Jac91].

5.3 Specific Aspects of PCA

5.3.1 Standardized Variables and Correlation Matrix

If the original variables are expressed in different scales or if their variances differ widely, variables with a large absolute value range risk to dominate the modeled correlation because they seem to contribute a lot to the variability. As a solution, the original variables can be standardized to zero mean and unit variance: $(x_i - \bar{x}_i)/\sigma_i$ for $i = 1, \dots, p$. The covariance of these standardized variables is identical to the correlation matrix of the original data.

The relationship between the elements of the sample covariance matrix $\mathbf{S}$ and the elements $r_{i,j}$ of the **sample correlation matrix R** is as follows:

$$r_{i,j} = \frac{s_{i,j}}{\sqrt{s_{i,i}s_{j,j}}} = \frac{s_{i,j}}{\hat{\sigma}_i\hat{\sigma}_j}$$

There is no one-to-one mapping between eigenvalues, eigenvectors, and scores calculated for the covariance matrix and the correlation matrix. Also, the number of equal eigenvalues may differ. What remains the same are the number of zero eigenvalues and the value of the T^2 statistic calculated from all PCs. The asymptotic distribution of the Q statistic is still valid, but, of course, the results differ.

5.3.2 Covariance Matrix Estimation

The accurate estimation of the covariance or correlation matrix is an essential prerequisite to determine appropriate PCs. For this reason, the training data must be representative for observations under normal conditions, which is the in-control state in the terminology of change detection. Outliers and anomalies in the training data may bias the covariances quite strongly. As a consequence, the eigenvalues, PCs, and test outcomes of PCA will be inadequate. If it cannot be guaranteed that the training data is free of outliers and anomalies, we can try to clean the data by removing extreme observations. However, the result may be biased if high values are completely neglected in the estimation. Another possibility to cope with outliers is to use covariance estimators which are robust, such as M-estimators and the mean square successive differences (MSSD).

In the case of **M-estimators** (see [Jac91, chap. 16.5.2] and [Seb84, chap. 4.4.3]), the observations are multiplied by weights $w_1(d_t)$ and $w_2(d_t^2)$ when calculating the sample mean and covariance matrix respectively:

$$\overline{\mathrm{x}} = \frac{\sum_{t=1}^{n} w_1(d_t)\mathrm{x}_t}{\sum_{t=1}^{n} w_1(d_t)}$$

$$\mathbf{S} = \frac{1}{n}\sum_{t=1}^{n} w_2(d_t^2)(\mathrm{x}_t - \overline{x})(\mathrm{x}_t - \overline{x})'$$

The weights depend on the Mahalanobis distance d_t between the observation x_t and the mean $\overline{\mathrm{x}}$:

$$d_t = \left\{(\mathrm{x}_t - \overline{\mathrm{x}})'\mathbf{S}^{-1}(\mathrm{x}_t - \overline{\mathrm{x}})\right\}^{1/2}$$

A possible choice for the weight functions is the Cauchy distribution:

$$w_1(d_t) = w_2(d_t^2) = \frac{p+1}{1+d_t^2}$$

As the Mahalanobis distance depends on the sample mean and covariance matrix, the estimation must be performed iteratively until the estimated values of $\overline{\mathrm{x}}$ and $\mathbf{S}$ converge. The classical sample covariance matrix and mean vector may serve as initial values. From the M-estimator covariance matrix, we can derive the elements of the associated correlation matrix as explained in Section 5.3.1.

Li [Li04] makes use of weights that depend on the residuals $\epsilon_t = \mathrm{x}_t - \hat{\mathrm{x}}_t$, where $\hat{\mathrm{x}}_t$ is obtained from the retained PCs (see Section 5.2.4). The weight applied to the i-th original variable is $\left\{1 + (\frac{\epsilon_{t,i}}{c})^2\right\}^{-1/2}$. The parameter c controls how fast the weights decrease with increasing residuals.

As the usual estimation of covariances based on the error products $(\mathrm{x}_t - \overline{x})(\mathrm{x}_t - \overline{x})'$ $(t = 1, \ldots, n)$ is very sensitive to outliers, Holmes and Mergen [HM93] propose an alternative covariance estimator using **mean square successive difference** (MSSD, cf. Section 4.5):

$$s_{i,j} = \frac{1}{2(n-1)} \sum_{t=2}^{n} (x_{t,i} - x_{(t-1),i})(x_{t,j} - x_{(t-1),j})$$

This estimator is more robust with respect to outliers and changes in the sample mean $\bar{\mathrm{x}}$. On the other hand, it is sensitive to serial correlation in the original variables.

5.3.3 Incremental PCA

A general problem of calculating sample covariances is that the observations are assumed to be independent. In the case of multivariate measurement time series, this condition is usually not fulfilled. Furthermore, dynamic changes of the normal behavior may invalidate the modeled correlation structure. As a consequence, batch-mode PCA requires to recalculate the PCs from time to time, using recent observations as training data. These limitations can be overcome by using **incremental PCA** with moving estimators as explained in the following.

Like the exponentially weighted moving average and variance estimators (EWMA, EWMV) for univariate time-series analysis (cf. Section 4.3.2), exponentially weighted moving covariance estimators can be used for incremental eigenanalysis and PCA [HMM98, LC02]. As in the univariate case, moving covariance estimators provide one-step-ahead forecast values which are quite robust with respect to serial correlation. In addition, the smoothing effect ensures that bias caused by sporadic outliers in the observations vanishes away over time.

Similarly to EWMV, an **exponentially weighted moving covariance** (EWMC) can be defined as follows $(i, j = 1, \ldots, p)$ [LC02]:

$$\begin{aligned} s_{i,j,t} &= \rho(x_{t,i} - \bar{x}_i)(x_{t,j} - \bar{x}_j) + (1-\rho)s_{i,j,(t-1)} \\ \Leftrightarrow \quad \mathbf{S}_t &= \rho(\mathrm{x}_t - \bar{\mathrm{x}}_t)(\mathrm{x}_t - \bar{\mathrm{x}}_t)' + (1-\rho)\mathbf{S}_{t-1} \end{aligned}$$

Li et al. [LYVCQ00] derive a recursive formula for the calculation of the correlation matrix which includes an additional corrective term reflecting that a new estimate of the mean is available with every recursion. An equivalent formula can be used for the covariance matrix:

$$\mathbf{S}_t = \rho(\mathrm{x}_t - \bar{\mathrm{x}}_t)(\mathrm{x}_t - \bar{\mathrm{x}}_t)' + (1-\rho)\mathbf{S}_{t-1} + \Delta\bar{\mathrm{x}}_t\Delta\bar{\mathrm{x}}_t'$$
$$\text{with} \quad \Delta\bar{\mathrm{x}}_t = \bar{\mathrm{x}}_t - \bar{\mathrm{x}}_{t-1}$$

Note that the EWMSV estimator of Eyvazian et al. [ENV07] (cf. Section 4.5) is included in the above equations for $i = j$. As a consequence of the corrective term, the latest estimate of the mean $\bar{\mathrm{x}}_t$, which is assumed to be the most accurate, is adopted as basis of the covariance estimation. Using EWMA for estimating the mean, we get:

$$\begin{aligned} & \bar{\mathrm{x}}_t = \rho\mathrm{x}_t + (1-\rho)\bar{\mathrm{x}}_{t-1} \\ \Rightarrow \quad & \Delta\bar{\mathrm{x}}_t = \bar{\mathrm{x}}_t - \bar{\mathrm{x}}_{t-1} = \rho\left(\mathrm{x}_t - \bar{\mathrm{x}}_{t-1}\right) = \frac{\rho}{1-\rho}\left(\mathrm{x}_t - \bar{\mathrm{x}}_t\right) \\ \Rightarrow \quad & \mathbf{S}_t = \left(\rho + \left(\frac{\rho}{1-\rho}\right)^2\right)(\mathrm{x}_t - \bar{\mathrm{x}}_t)(\mathrm{x}_t - \bar{\mathrm{x}}_t)' + (1-\rho)\mathbf{S}_{t-1} \end{aligned}$$

As can be seen, the corrective term increases the influence of the new observation on the estimated covariance matrix. However, the effect of the corrective term is small and negligible for small ρ.

The M-estimator weights can be easily integrated into the formulas of the moving estimators in order to increase the robustness against outliers. Li proposes such a combination of M-estimators and incremental PCA, using an approximation of the covariance matrix which is calculated from the retained PCs only [Li04]. This allows the author to reduce the complexity of updating the retained PCs as well. Instead of calculating the eigenvectors of the $p \times p$ covariance matrix, he derives the retained PCs from the eigenvectors of a smaller $(k+1) \times (k+1)$ matrix, where k is the number of retained PCs.

If MSSD is to be used for covariance estimation, we can construct the following moving estimator:

$$s_{i,j,t} = \rho\frac{(x_{t,i} - x_{(t-1),i})(x_{t,j} - x_{(t-1),j})}{2} + (1-\rho)s_{i,j,(t-1)}$$

However, we have not found any related work where this estimator is used.

Common to all moving estimators is the smoothing parameter ρ. A formula to calculate the optimal value does not exist, thus an appropriate value of ρ has to be found by experimentation.

5.3.4 Effect of Systematic Changes on PCA

If applied to multivariate time series, PCA ignores the temporal order of the observations and therefore does not consider any temporal relationships in the data. Variables exhibiting systematic changes, such as trend or seasonal variation, are considered as contributors of a large proportion of variability and will be reflected in the first PCs. If multiple of the original variables are affected by the same systematic changes, PCA allows separating the non-stationary part from the remaining variability in the data. Precisely this phenomenon is exploited by many existing PCA-based traffic anomaly detection approaches presented in Section 5.4.

We illustrate the relationship between systematic changes and the PC scores obtained by PCA with an example. Assume that there are two original variables, such as the number of packets and the number of bytes observed on the same link, which are both affected by the same kind of seasonal variation. In particular, we usually observe that the packet count and the byte count depend on the time of day. As the seasonal variation contributes a large proportion of the variability, the two variables appear to be highly correlated. We can also interpret it the other way round: because of the correlation, a change in one of the variables is likely to appear in the other variable as well.

If PCA is applied, the first PC will reflect the correlation between the two variables. The resulting PC score time series contains the seasonal variation which is common to both metrics. The other PC score contains the remaining variability in the variables. Ideally, the corresponding time series resembles the output of a stationary random process unless any further sources of systematic changes exist.

In the general case with more than two variables, we can retain those PCs which exhibit non-stationary behavior over time (these PCs are typically among the first). The residual variability not covered by these PCs will be more or less free of systematic changes. Hence, the Q statistic and the T_H^2 statistic can be used to detect significant values in the residual PCs, indicating a change in the correlation structure of the original variables (see Section 5.2.4).

As mentioned in Section 5.2.4, the T^2-test can be applied to the retained PCs in order to detect outliers. Just like Hotelling's T^2-test presented in Section 5.2.3, this test relies on the assumption that observations at different points in time are independent and identically distributed. This assumption is obviously not fulfilled if the original variables exhibit common systematic changes. As a remedy, we could try to eliminate systematic changes and serial correlation in the scores and the T^2 statistic using time-series models and forecasting techniques as presented in Section 4.3.

5.4 Existing Traffic Anomaly Detection Approaches Using PCA

Shyu at al. [SCSC03] apply PCA to 34 numeric features of the KDD cup 1999 datasets [KDD99] which are derived from the 1998 DARPA intrusion detection evaluation [DAR01]. Every observation is associated to a connection and not to a time interval. The T^2 statistic is calculated separately for the first (major) PCs accounting for half of the variability in the original data, and for the remaining (minor) PCs. Anomalies are detected using empirical thresholds for T^2. Shyu at al. also discuss the problem of outliers in the training data which may distort the estimation of mean and covariances. Therefore, they apply multivariate trimming by removing the most extreme observations from the training data before calculating the sample mean and correlation matrix.

Lakhina et al. apply PCA to multivariate time series of byte, packet, or flow counts measured on different links [LCD04b] or for different traffic aggregates [LPC$^+$04, LCD04a]. These aggregates, called origin-destination flows (OD flows), consist of the traffic transfered from one ingress PoP (point of presence) to an egress PoP of the network. Each time-series value accounts for a time interval of 5 or 10 minutes. In the first work [LPC$^+$04], Lakhina et al. apply PCA to byte counts of OD flows in order to examine the resulting PCs. Using scree plots, it is shown that a small number of PCs contributes most of the variability, regardless of whether the original variables are standardized or not. Furthermore, the authors distinguish three types of PC scores: PC scores with diurnal patterns, PC scores showing spikes, and PC scores resembling Gaussian noise. In another paper [LCD04b], Lakhina et al. detect traffic anomalies in byte counts of individual links with help of the Q statistic. In yet another publication [LCD04a], anomalies are detected in byte, packet, and flow counts of OD flows by examining the residuals as well as the retained PCs using the Q statistic and the T^2 statistic, respectively. Moreover, the authors show that attacks, scans, outages, and other events result in different types of anomaly patterns.

In another work [LCD05], Lakhina et al. apply PCA and residual analysis to entropy values reflecting the distribution of source and destination addresses and transport ports in the OD flows. These entropy values are very sensitive to distributional changes caused by attacks, scans, flash crowds etc. Under the co-authorship of Lakhina, Li et al. adopt this approach to detect anomalies in traffic aggregates that result from hashing the flow keys to the bins of sketches [LBC$^+$06]. Using multiple sketches with different

Tab. 5.1: PCA-based traffic anomaly detection approaches

Reference	Traffic aggregates	Traffic metrics	Deployed statistic
Shyu [SCSC03]	1 (sniffer)	34 (different connection and host-based features)	T^2, T_H^2
Lakhina [LCD04b]	49, 41 (links)	1 (byte count)	Q
Lakhina [LCD04a]	121 (OD-flows)	1 (byte, packet, or flow count)	Q, T^2
Lakhina [LCD05]	121, 484 (OD-flows)	4 (entropy of source and destination IP addresses and ports)	Q
Li [LBC+06]	121, 484 (hashed flow keys)	4 (entropy of source and destination IP addresses and ports)	Q
Chatzigiannakis [CPA09]	6, 10 (links)	2 (byte and packet count)	Q
Brauckhoff [BSM09]	1 (link)	28 (byte, packet, and flow count, source and destination IP address entropy, numbers of distinct source and destination ports, separately for TCP/UDP and incoming/outgoing traffic)	Q

hash functions, the authors are able to identify the responsible flows in the case of an alarm.

Lakhina does not discuss the effect of systematic changes although all of the applied statistical tests rely on the assumption that observations at different points in time are independent and identically distributed. Internet traffic is strongly affected by daily variation, which results in a certain number of PC scores with diurnal patterns [LPC+04]. We assume that Lakhina actually eliminates daily variation by including these PCs in the set of retained PCs.

Chatzigiannakis et al. deploy PCA to reduce the dimensionality of multivariate data measuring multiple metrics on multiple links in the network [CPAM06, CPA09]. The original variables are standardized to remove the influence of different units and scales. Apart from minor differences, the approach is identical to the one proposed by Lakhina.

Ringberg et al. try to reproduce the results obtained by Lakhina and find that it is difficult to obtain good anomaly detection results [RRSD07]. One reason is that the outcome depends very much on the number of retained PCs. Furthermore, anomalous values in the training data are made responsible for inappropriate PCs. Ringberg et al. also criticize that it is very difficult do identify the responsible traffic aggregates after having detected an anomaly.

Brauckhoff et al. [BSM09] argue that the main problem of Lakhina's approach is serial correlation in the measurement data which cannot be taken into account by conventional PCA. As a solution, the authors use PCA to determine the coefficients of the Karhunen-Loeve (KL) expansion which reflects correlation between different variables as well as between adjacent time intervals. The approach is applied to multivariate time series composed of the byte, packet, and flow counts, the source and destination IP address entropy, as well as the unique source and destination IP address counts measured on a peering link for incoming and outgoing TCP and UDP traffic. Hence, in contrast to Lakhina, Brauckhoff et al. do not analyze measurement data from distributed observation points but only from a single router. All variables are standardized to zero mean and unit variance before applying PCA.

Table 5.1 compares the different approaches with respect to the number of considered traffic aggregates, the evaluated traffic metrics, and the deployed test statistic. If there appear two numbers in the second column, they refer to different datasets evaluated in the paper. Multiplying the number of aggregates with the number of metrics results in the total number of original variables. In the case of Brauckhoff et al., the result must be multiplied by the number of adjacent time intervals considered in the KL expansion.

5.5 Concluding Remarks on PCA

As described in the preceding sections, PCA allows modeling the inherent correlation structure of multivariate data. Hence, certain kinds of anomalies and outliers can be easier detected in the PC scores than in the original variables. Furthermore, highly dimensional data can be approximated by a much smaller number of PCs. Finally, PCA allows detecting and isolating seasonal variation that effects multiple of the original variables. In the following, we summarize the critical aspects regarding the applicability of PCA.

PCA requires knowledge of the means and covariances of the original variables. Typically, these properties are not known a priori and must be estimated. In Section 5.3.2, we discussed the difficulties concerning the covariance estimation. In the case of batch-mode PCA, the covariance matrix is estimated from training data which needs be representative for normal observations and free of outliers and anomalies. The computational complexity of this estimation is in the order of $O(p^2n)$, with p being the number of variables and n the number of observations in the training data. The complexity of calculating the eigenvalues and eigenvectors, for example with QR factorization [GL96], is $O(p^3)$ per iteration. Thus, determining the PCs is complex for large p, which must be taken into account if we think of updating the PCs at a regular basis in order to adapt to changes of the normal behavior.

Batch-mode PCA is based on mean and covariance estimators which rely on independent observations. However, this assumption is typically not fulfilled in the case of traffic measurement time series. Hence, the estimates are biased and do not necessarily describe the actual statistical properties of the underlying random process.

As an alternative to batch-mode PCA, incremental PCA makes use of exponentially weighted moving estimators for the mean and covariance matrix. These estimators are expected to provide robust one-step-ahead forecast values. Hence, we can determine a set of PCs which correspond to the expected correlation structure of the next observation in the time series. A drawback is that the continuous re-estimation of the covariance matrix entails frequent recalculations of the eigenvalues and eigenvectors as well. Furthermore, we must be aware of the fact that score values obtained at different instances of time result from different PC transformations and therefore should not be directly compared with each other.

If PCA is used to reduce the dimensionality, only a small subset of the PCs contributing most of the variability is retained. However, the decision on the appropriate number of retained PCs is not trivial. A couple of stopping rules exist, yet they may lead to very different results. Hence, the most appropriate number must usually be found empirically given the data to be analyzed.

The sensitivity of PCA to anomalies in the training data and the difficulty to find the best number of retained PCs are also reported by Ringberg et al. as the main limitations of Lakhina's anomaly detection approach [RRSD07]. Furthermore, Ringberg et al. mention the difficulty to infer the subset of traffic aggregates (i.e., links or OD-flows) which are responsible for an alarm. This problem can be generalized to the question in which of the original variables an anomaly has occurred. Lakhina looks for the original variables which contribute the most to an anomalously large Q or T^2 value. However, this approach disregards the fact that simultaneous small deviations in many of the original variables may also sum up to large values in these statistics.

Changes and anomalies can be detected in the individual PC scores as well as in the T^2, T^2_H, and Q statistics using statistical tests or control charts. These tests are parametric, which means that knowledge about the probability distribution is required to calculate critical values and control limits. Accurate critical values can be derived if the original variables are multivariate normally distributed. However, as this assumption is usually not fulfilled for traffic measurement data, appropriate thresholds can only be approximated or found empirically.

In addition to following an unknown statistical distribution, the measurement variables are even not stationary in many cases. As discussed in Section 5.3.4, systematic changes in the original time series remain present in at least some of the PC scores. Hence, we have to be careful when applying T^2, Q, and T_H^2-tests since these rely on the assumption of independent and identically distributed observations. The score time series of the residual PCs should be free of systematic changes in order to use the T_H^2-test or the Q-test. In the next chapter, we assess the effect of long-range dependence on PCA, which is assumed to have a similar impact as non-stationarity.

Serial correlation in the residual PCs can be reduced with help of the KL expansion, as proposed by Brauckhoff et al. [BSM09]. However, this approach multiplies the number of variables and thus increases the complexity of PCA significantly. Another possibility to adapt to temporal changes without increasing the dimensionality of the data offers incremental PCA as mentioned in Section 5.3.3.

Regarding traffic anomaly detection, it is difficult to predict under which conditions PCA applied to multivariate time series yields better anomaly detection results than modeling the temporal behavior of univariate time series. If anomalies become manifest in significant deviations from usual seasonal changes in the data, modeling the temporal behavior as described in Chapter 4 are more promising. However, subtle changes in the correlation structure which do not affect the temporal behavior of individual variables are expected to be detectable with PCA only. Therefore, whether time-series analysis or PCA is better depends very much on the types of anomalies that occur in the analyzed data.

In Chapter 8, we apply batch-mode PCA as well as incremental PCA to multivariate traffic measurement time series. We analyze the resulting time series of individual PC scores as well as the T^2 and T_H^2 statistics and deploy change detection methods to detect traffic anomalies. The results are compared to the changes detected in the residuals of univariate forecasting methods in order to assess if PCA is better or not.

6. INTERNET TRAFFIC CHARACTERISTICS

6.1 Introduction

The long-term evolution of Internet traffic is characterized by a sustained growth which can be assumed to be approximately exponential [WP98, Odl03]. At the time scale of hours, the traffic shows a seasonal pattern synchronized with the time of day, which can be explained by different network usages at daytime and at nighttime. Traffic dynamics at even smaller time scales of minutes, seconds, and milliseconds exhibit self-similar structures and long-range dependence in certain traffic statistics [LTWW93].

In the given context, self-similar means that traffic statistics appear to be invariant with respect to a change in the time scale. This is in contrast to the Poisson traffic model used to dimension and evaluate telephone networks. Hence, when the self-similar structure of traffic in data networks was discovered [LTWW93], it provided a cogent explanation why the true performance of data networks fell short of the theoretical performance calculated under the assumption that the traffic model of telephone networks was still valid.

Section 6.2 summaries the main aspects of traffic modeling in telephone networks. Section 6.3 then gives an overview on self-similar structures and long-range dependence in Internet traffic. For a more detailed introduction to this topic, we refer to Stallings' textbook on high-speed networks [Sta01].

Long-range dependence conflicts with the theoretical foundations of the univariate and multivariate anomaly detection methods presented in Chapters 3 and 5 which assume that observations are independent. As discussed in Section 4, time-series models and forecasting techniques help to reduce temporal dependencies in measurement time series. If principal component analysis (PCA) is used to model the correlation structure in multivariate time series (see Chapter 5), we can exploit the fact that temporal dependencies usually occur simultaneously in multiple measurement variables. As described in Section 5.3.4, such temporal dependencies are then mapped to a small number of principal components.

In Section 6.4, we assess how long-term dependence in the traffic measurement data may affect the detection of anomalies based on time-series forecasting and PCA. Section 6.5 closes this chapter with some concluding remarks about non-stationarity and long-range dependence.

6.2 Traffic Models in Telephone Networks

In circuit-switched telephone networks, call arrivals from a single source can be approximated by a Poisson process where the interarrival time between two subsequent calls is negative-exponentially distributed. Due to the memoryfree property of the Poisson process, the numbers of call arrivals in non-overlapping equally-spaced time intervals are independent and identically distributed according to the Poisson distribution. Hence, the Poisson process is a pure random process as described in Section 4.2, which means that its output is stationary and does not exhibit any serial correlation.

An important property of the Poisson traffic model is that the aggregation of call arrivals from many sources is again a Poisson process with larger expectation value but smaller variability (measured by the coefficient of variation). At high levels of aggregation, small delay or blocking probabilities can be guaranteed with a system dimensioned to handle the mean call arrival rate plus a relatively small safety margin. Hence, higher utilization of system resources can be achieved with increased traffic aggregation, which is known as the "economy of scale" effect.

6.3 Self-Similar Traffic

Packet arrival processes in data networks have different properties than call arrival processes in telephone networks. Intuitively, the arrival times of packets originating from a single source are not independent but correlated to each other because applications often transmit data blocks which exceed the maximum packet size and therefore need to be partitioned into a sequence of multiple packets. As a result, packet arrivals tend to occur in bursts or "packet trains" as observed by Jain and Routhier [JR86].

From the observation of correlated packet interarrival times, we can deduce that the number of packets and bytes measured in non-overlapping equally-spaced time intervals should be correlated as well. The analysis of traffic captured in different networks has shown that packet and byte count time series exhibit long-range dependence and self-similar structures over a wide range of time scales from milliseconds to minutes (e.g., [LTWW93, FGW98]). The reason is that bursts of packets often occur in a short distance, which makes them look like a single long burst at larger time scales. As a result, packet and byte count time series look similar for different time interval lengths.

Formally, the statistical properties of a self-similar process only change in scale when the time scale changes. The discrete time definition of self-similarity compares the process output $\{X_t | t = 1, 2, \ldots\}$ to the m-aggregated time series $\{X_k^{(m)} | k = 1, 2, \ldots\}$ which is defined as

$$X_k^{(m)} = \frac{1}{m} \sum_{t=km-(m-1)}^{km} X_t$$

Thus, $X_k^{(m)}$ is the average of m consecutive output values $X_{km-(m-1)}$ to X_{km}. The underlying stochastic process is **exactly (second-order) self-similar** with parameter β $(0 < \beta < 1)$ if for all $m > 1$:

$$Var\left[X_k^{(m)}\right] = \frac{Var\left[X_t\right]}{m^\beta} \quad \text{and} \quad Corr\left[X_k^{(m)}, X_{k+\tau}^{(m)}\right] = Corr\left[X_t, X_{t+\tau}\right]$$

$Corr[X_k^{(m)}, X_{k+\tau}^{(m)}]$ and $Corr[X_t, X_{t+\tau}]$ are the autocorrelation functions of $X_k^{(m)}$ and X_t, respectively. If the first condition holds approximately and if the two autocorrelation functions are asymptotically equivalent for large m and large k, the process is asymptotically (second-order) self-similar. The parameter β is related to the so-called Hurst parameter $H = 1 - \frac{\beta}{2}$.

For a pure random process (see Section 4.2), it is $Var[X_k^{(m)}] = \frac{Var[X_t]}{m}$. In the case of a self-similar process, the variance of $X_k^{(m)}$ decays more slowly with increasing m, which implies that the time-series values are correlated. Furthermore, as the autocorrelation does not vanish with aggregation in time, the process exhibits long-range dependence or it is non-stationary.

Long-range dependence refers to the autocovariance of a stationary process. A stationary process is long-range dependent if its autocovariance decays hyperbolically with lag τ:

$$Cov\left[X_t, X_{t+\tau}\right] \sim |\tau|^{-\beta} \quad \text{for } \tau \to \infty \text{ , } 0 < \beta < 1$$

In the above equation, β is the same parameter as used in the previous equation. As we can see, β determines the degree of long-range dependence or persistence. The sum $\sum_{\tau=0}^{\infty} Cov\left[X_t, X_{t+\tau}\right]$ is infinite.

In comparison, the autocovariance of a short-range dependent process, such as an ARMA process (see Section 4.3.1), decays exponentially (i.e., $Cov\left[X_t, X_{t+\tau}\right] \sim a^{|\tau|}$ with $0 < a < 1$) and results in a finite sum of autocovariances. In this case, $Corr[X_k^{(m)}, X_{k+\tau}^{(m)}] \to 0$ for $\tau \neq 0$ and $m \to \infty$.

As mentioned at the beginning of this paragraph, traffic sources in data networks usually emit bursts of packets. This behavior can be related to the self-similar structure in the measurement time series of aggregated traffic by modeling sources according to a phase type model (or "packet train" model) with alternating active and inactive phases. During active phases, the source emits packets at a constant or variable rate; during inactive phases, the source is silent. Aggregating multiple of these sources leads to approximately self-similar traffic if the duration of at least one of the two phases follows a

heavy-tailed distribution [CB97]. The distribution of a random variable X is heavy-tailed if:

$$\Pr[X > x] \sim \frac{1}{x^{\alpha}} \quad \text{for } x \to \infty \ , \ \alpha > 0$$

Heavy-tailed distributions have infinite variance, which results in a very high variability. The expectation value may be infinite as well. An example for a heavy-tailed distribution is the Pareto distribution.

Self-similar structures in network traffic have first been discovered by Leland et al. [LTWW93] through the examination of byte and packet count time series measured in local-area networks (LANs) between 1989 and 1991. As a possible reason, the authors mention the aggregation of traffic from many sources generating packet bursts with heavy-tailed length and heavy-tailed gaps between bursts. In a later publication, the authors validate this assumption by evaluating the active and inactive times for different source-destination pairs in a LAN [WTSW97].

Paxson and Floyd [PF95] examine session, connection, and packet arrivals of different applications in wide-area network (WAN) traffic and observe that the Poisson assumption holds for the arrivals of user-initiated Telnet and FTP sessions. On the other hand, Telnet packet arrivals and FTP data connection arrivals follow a heavy-tailed distribution. Moreover, the authors confirm the existence of long-range dependence in the traffic measurement data.

Crovella and Bestavros [CB97] analyze the causes of self-similar structures in web traffic. The authors find that self-similarity is the result of the superposition of web connections with heavy-tailed length and heavy-tailed thinking times of the users The heavy-tailed distribution of the connection lengths is related to the heavy-tailed distribution of files sizes on the web servers.

With help of discrete wavelet transform, Feldmann et al. [FGWK98, FGW98] analyze the structure of WAN and LAN traffic at various time scales and discover significant differences at small time scales below a few hundred milliseconds. The reason for these differences is the round trip time (RTT), which is typically in the range of a few hundred milliseconds in WANs and much smaller in LANs. Feldmann et al. discover that WAN traffic exhibits a multifractal structure at small time scales, which means that the self-similar structure varies over time. Temporary congestion in the network as well as TCP retransmissions after timeout can be made responsible for these temporal changes [HFW01].

Uhlig and Bonaventure [UB01] examine *NetFlow* data collected in an ISP network and determine that the time series of the number of external source IP addresses exhibits similar self-similar structures as the byte count time series. Hence, the authors conclude that the number of sources may have an effect on the self-similar structure of Internet traffic, besides the heavy-tailed traffic volume generated by a single source. However, the authors' conclusion could be biased by the fact that the evaluated time series lasts for six days and shows non-stationarity due to daily variation.

Mori et al. [MKHS10] analyze the statistics of flows carrying video data from video sharing websites. Today, these flows make up a significant proportion of Internet traffic in terms of traffic volume (i.e., number of bytes and packets). The authors show that the flow length approximately follows a truncated Pareto distribution combined with a second Pareto distribution accounting for a small set of largest flows. The truncation is related to the capacity limitations for videos offered by nonpaying members of the video portals.

In summary, we can say that Internet traffic exhibits asymptotically self-similar structures and long-range dependence which can be related to the heavy-tailed length of packet bursts, TCP connections, user sessions, object sizes on web servers etc. In the next section, we discuss possible implications for traffic anomaly detection.

6.4 Implications for Traffic Anomaly Detection

The statistical change detection methods presented in Chapter 3 rely on the assumption of independent and identically distributed observations. In Chapter 4, we present time-series forecasting as a possible solution to model and remove temporal dependencies in traffic measurement data. In the following, we assess whether this approach is still practical if the analyzed time series exhibit long-range dependence.

Chatfield denotes that "it is generally more difficult to get good estimates of some parameters of [a long-range dependent process], notably the mean, but it is usually possible to make better forecasts" (see [Cha03, chap. 13.5.7]). The last part of his statement is intuitively clear since it is easier to make a prediction if past observations are known to be correlated to future ones.

Long-range dependence can be modeled with fractional time-series models, such as the fractional ARIMA (ARFIMA) model. The ARFIMA(p,d,q) model generalizes the ARIMA(p,d,q) model (see Section 4.3.1) by allowing parameter d to be a non-integer. The implied fractional differencing is based on the backwards shift operator B and the binomial expansion:

$$\nabla^d X_t = (1-B)^d X_t = \left(\sum_{k=0}^{\infty} \binom{d}{k} (-B)^k \right) X_t$$

For $0 < d \leq \frac{1}{2}$, an ARFIMA process is stationary and long-range dependent with Hurst parameter $H = d + \frac{1}{2}$.

Several studies have investigated the advantages and disadvantages of ARFIMA models over usual AR(I)MA models which do not reflect long-range dependence. For example, Ray [Ray93] evaluates the theoretical forecasting inaccuracies which occur if an AR(p) model is used to predict the output of a stationary, long-range dependent ARFIMA process. In this study, high-order AR models provide useful long-term forecast values.

Smith and Yadav [SY94] use ARIMA(p,1,0) models to calculate forecast values for non-stationary ARFIMA processes (i.e., ARFIMA(0,d,0) with $d > 0.5$). The simulation shows that ARIMA forecasts are only slightly worse, except for the one-step-ahead forecasts where the mean square error increases by 15 percent for $d = 0.7$.

Crato and Ray [CR96] examine forecasting for the output of stationary ARFIMA processes ($d < 0.5$). When ARFIMA and ARMA model parameters are estimated from the same number of observations, the resulting ARMA model delivers equivalent or better forecast values than the ARFIMA model in most of the cases, despite of the inaccurate model assumption. This result can be explained by the difficulty to estimate accurate parameters for ARFIMA models with a limited set of observations.

We have not found any similar evaluations regarding the forecasting performance of exponential smoothing, which is one of the preferred methods for our purposes as discussed in Section 4.7. Exponential smoothing provides optimal forecast values for an ARIMA(0,1,1) process (cf. Section 4.3.2). Regarding the results reported for other ARIMA models, we assume that long-range dependence does not have a significant negative effect on the one-step-ahead forecast values of exponential smoothing either.

Roughan [Rou09] examines the impact of long-range dependence on the detection of anomalous time-series values. Therefore, he generates long time series of fractional Gaussian noise and substitutes a couple of time-series values by the output of a uniformly distributed random variable. Then, he applies an anomaly detection mechanism which determines the deviation of the current time-series value from the moving average of a sliding window. The interesting outcome of his simulation is that the stronger the long-range dependence is, the better the anomaly detection performs. Roughan derives an analytical explanation: the mean square deviation of the moving average is smaller for strongly correlated time-series values, which allows using narrower thresholds and thus facilitates the detection of outliers. Although Roughan's evaluation is based on simplifying assumptions, it shows that long-range dependence can be advantageous for the detection of anomalies in time series.

The publications presented so far concern forecasting and anomaly detection in univariate time series. The situation is different for the PCA-based detection of deviations from the usual correlation structure in multivariate measurement data described in Chapter 5. The calculation of the principal components (PCs) relies on covariance estimates, such as the sample covariance matrix or the M-estimator covariance matrix in the case of batch-mode PCA. Long-range dependence in the measurement data makes it difficult to obtain good covariance estimates because the estimators converge very slowly to the true parameter values of the underlying process.

In Section 5.3.4, we discuss the effect of systematic changes on PCA and point out that PCA can help to separate seasonal variation observed in multiple original variables from the remaining variability in the data. The same applies if the influence of long-range dependence on different measurement variables occurs in a correlated way. Actually, certain dependencies between different variables, such as the number of packets and the number of bytes, exist irrespective of long-range dependence or systematic changes. Hence, long-range dependence and systematic changes can be handled in the same way by having a critical look at the sample autocorrelation functions of the PCs and PCA-related statistics in order to assess the influence of temporal dependencies in the original variables.

6.5 Concluding Remarks on Anomaly Detection in Internet Traffic

As mentioned in the introduction of this chapter, Internet traffic is characterized by a positive long-term trend, daily variation, and long-range dependence. All these effects collide with the assumption of independent and identically distributed observations which is the base for the statistical methods presented in Chapters 3 and 5. However, the different effects are linked to different time scales. Hence, whether we need to primarily cope with non-stationarity or long-range dependence in the measurement data depends on the considered interval length.

The evaluations presented in Part III are based on traffic measurement time series with an interval length of five minutes. If we consider a sequence of several time-series values, the influence of daily variation is visible as a local trend. Thus, correlation between distant observations mainly goes back to systematic changes while long-range dependence may explain some of the remaining correlation. On the other hand, considering the correlation with the most recent observations is the most important for calculating one-step-ahead forecast values (cf. Section 4.3.2). Therefore and because of the results published for various AR(I)MA models (see Section 6.4), we expect that robust forecasting methods, such as exponential smoothing, work fine with the given time series.

As discussed in the preceding section, long-range dependence is expected to have a similar effect as non-stationarity when PCA is applied to multivariate traffic measurement data. In comparison, we assume that the effect of seasonal variation due to different day and night regimes will be much larger than the effect of long-range dependence.

Roughan [Rou09] has shown that long-range dependence may be beneficial for the anomaly detection. Additional studies would be useful to confirm whether this observation holds under different assumptions, for example in the case of anomalies which are superimposed on long-range dependent background traffic in an additive manner.

Part III

APPLICATION AND EVALUATION

7. EVALUATION SETUP AND ANALYZED DATASET

7.1 Introduction

In the next chapters, we apply and evaluated different residual generation and change detection methods to multivariate traffic measurement time series. These time series are derived from flow data collected in real networks. All detected anomalies have happened in reality, which means that we do not use any simulated or synthetically generated attack traffic as done in some related publications. Although the analyzed flow data is historic and was collected in 2006, we assume that it is still quite representative for traffic in today's networks. In Section 7.2, we give some more details about the network where the flow records have been collected.

We store the flow records in a database which allows us to flexibly generate time series of different metrics and of different parts of traffic as explained in Sections 7.3 and 7.4. The studied residual generation and change detection methods have been implemented as *GNU Octave* functions as described in Section 7.6. This approach allows us to conveniently try out various parameter settings on the same dataset obtaining reproducible and directly comparable results.

Despite the fact that our evaluation is performed offline, the applied anomaly detection methods work in an online fashion, which means that they sequentially evaluate time-series value and eventually raise an alarm without considering future observations. Hence, the anomaly detection methods can be easily implemented as part of a real-time traffic analysis system. For example, we have integrated some of the detection methods as detection modules into the *TOPAS* framework presented in Section 2.4.2.

7.2 Analyzed Flow Dataset

The analyzed flow dataset was collected in the Gigabit backbone network of a regional ISP between September 7 and November 16, 2006. The operation area of the ISP covers parts of Saarland, Rhineland-Palatinate, Hesse (all federal states in Germany), Luxembourg, and Belgium. At measurement time, the offered services ranged from server hosting and colocation to VPNs and modem, ISDN, and DSL dial-in service. Customers were corporate clients, local carriers, roaming providers, and small and medium enterprises. The backbone connected ten points of presence (PoPs) and three peering

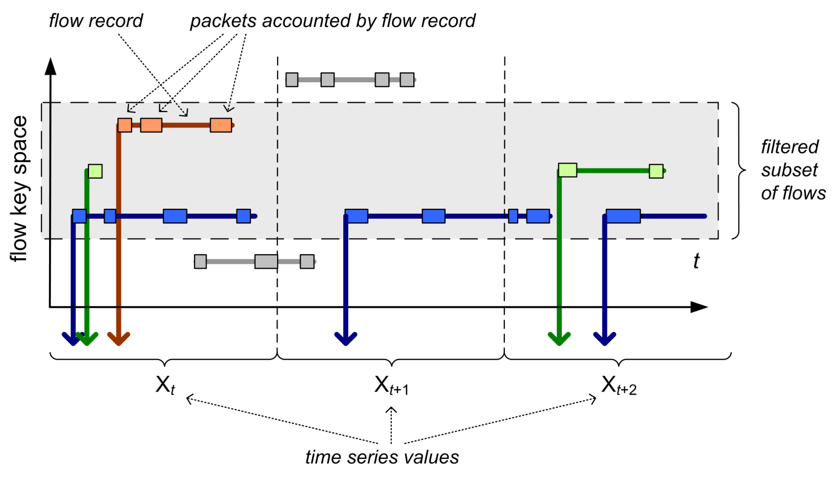


Fig. 7.1: Flow to time series conversion

points in Germany (DENIX, Frankfurt), Belgium (BNIX, Bruxelles), and Great Britain (LNIX, London).

The measurements were performed at a router using unsampled *Cisco NetFlow* version 5 with active and passive flow timeouts set to 150 seconds. The router exported the resulting flow records to a *flow-tools* collector [FR00]. The CryptoPAN [XFAM02] algorithm was applied for prefix-preserving IP address anonymization.

The *flow-tools* collector saved the *NetFlow* data in a file-based flow repository. For further analysis and conversion into time series, we imported the data into a *MySQL* database [MyS10].

7.3 Flow to Time Series Conversion

As explained in Section 2.2.2, flow-level measurements produce flow records with statistics about the observed packet flows. Flows are distinguished by flow keys, which typically consist of the IP quintuple. Each flow is reported in one or multiple flow records depending on the configured timeouts.

In order to convert the collected flow records into time series, we define a time grid with fixed interval length. Each flow record is associated to the time interval in which the first packet accounted for the flow has been observed. For a given time interval, the metric values are derived from those flow records whose start timestamps fall into the interval. Figure 7.1 illustrates the flow to time series conversion as well as the relationship to individual packets.

The described assignment of flow records to time intervals would lead to accurate traffic measurement values if start and end timestamps of each flow record fell into the same interval. In other cases, we partly account for traffic that was actually observed in later time intervals. To solve this problem, the packet and byte counts reported in a flow record could be proportionally distributed to the concerned intervals. However, as we do not know the exact arrival times and lengths of the packets, this distribution can only be a rough approximation of the real traffic.

We decide not to distribute record statistics over multiple intervals. Instead, we choose an interval length that is twice as long as the longest duration reported in the records in order to limit possible distortions. The longest flow duration is bounded by the active timeout of the flow metering process, which is 150 seconds for the analyzed dataset. Thus, the generated time series use an interval length of 300 seconds (i.e., five minutes). Five minute intervals have also be used in many existing publications (e.g., [LCD05]).

By applying a filter to the flow keys, time series can be determined for a subset of flows, which is illustrated by the gray box in Figure 7.1. For example, we can produce a time series for ICMP traffic or for TCP traffic from and to port 80, which is expected to be web traffic.

7.4 Traffic Metrics

We use the term 'metric' for a scalar measure of the traffic. Thus, if a single metric is determined for every time interval, the conversion of flow records results in a univariate time series. In the case of multiple metrics, we obtain a multivariate time series.

In order to be appropriate for online anomaly detection, the considered metrics must fulfil certain requirements:

- The computation of the metrics should be computationally inexpensive.
- The memory requirements of the computation should be constant or at least increase less than linearly with the number of flow records per time interval. Ideally, the calculation of a time-series value can be performed on a stream of flow records without storing every individual record.
- The metrics must be derived from the flow records of the given time interval only and must not depend on external data or flow information from other intervals.

If the above conditions are fulfilled, time-series values can be calculated efficiently and independently from each other.

The metrics that we consider can be classified into **volume metrics** and **cardinality metrics**. Volume metrics describe the traffic volume. In our evaluation, we use the total number of bytes, packets, and flow records counted in the given time interval. Volume metrics can be determined very easily by summing up the corresponding values of the flow records. Cardinality metrics indicate the numbers of distinct values observed for one of the flow keys or a combination of several flow keys. In the evaluation, we use the numbers of distinct source IP addresses, destination IP addresses, source ports, and destination ports. These cardinality metrics are useful for detecting scanning activities or distributed attacks because anomalously high values can be expected in such situations.

Calculating exact values for cardinality metrics requires a hash table with a distinct entry for every observed flow key value. Hence, the required memory is proportional to the number of distinct values observed. The number of distinct values is bounded above by the value range of the flow key. The problem is that this theoretic boundary may be very high. For example, the value range of an IP version 4 address field is $2^{32} = 42949672964$.

Instead of determining exact cardinality values, we can apply probabilistic counting methods which require a much smaller and constant amount of memory. Probabilistic counting methods have been proposed by Flajolet et al. [FM85, DF03, FFGM07] and Whang et al. [WVZT90]. Applications in the area of network traffic measurements have been discussed by Estan et al. [EVF03, KME05] and Cormode et al. [CMZ06]. These methods can be applied if the real-time calculation of the exact cardinality values is to costly. Note that our evaluation is based on exact values.

Another metric which can be easily obtained but does not belong to the volume and cardinality metrics is the average flow duration. The average flow duration is the average difference between the end and start timestamps of the flows. Including this metric, we consider a set of eight metrics in total.

Some existing approaches make use of entropy values for anomaly detection [LCD05, BMP07, TBSM09]. The entropy is a metric which describes the frequency distribution of the observed values of a flow key in a given time interval. In the case of the Shannon entropy, the entropy takes a value between 0 and $\log_2 L$, where L is the size of the value range (i.e., the number of possible values). Small entropy values indicate that a few flow key values appear much more frequently than others. For increasing entropy values, the frequencies are more and more uniformly distributed. If we know the range of typical entropy values, anomalously small values as well as anomalously large values are signs of traffic anomalies.

Before calculating the entropy, the frequency distribution must be determined for the given flow key. This requires a large hash table maintaining a counter for every observed flow key value. The calculation of the entropy itself is based on a quite complex formula, such as $H = -\sum_{i=1}^{L} q_i \log_2 q_i$ in the case of Shannon (with q_i being the relative frequency of the i-th value). In general, we expect that most entropy anomalies can also be detected in at least one of the cardinality metrics or volume metrics. Therefore, and because of the rather high complexity required to calculate entropy values, we do not use these metrics in our evaluation.

7.5 Relevant and Irrelevant Anomalies

We do not dispose of a ground truth for the given dataset, meaning that we do not have any context information about what was happening in the network at measurement time. However, if an anomaly is detected in a specific time-series interval, we can inspect the original flow records from which the time series values have been derived. By comparing the flow records with those measured in preceding intervals, it is usually possible to identify changes in the traffic which are responsible for the anomaly.

In order to compare different anomaly detection methods, we classify the detected anomalies as relevant or irrelevant from the point of view of the network administrator. An anomaly is considered relevant if it is caused by harmful or unwanted traffic or severe network problems, such as network and port scans, password guessing, and network outages. Otherwise, it is classified as irrelevant.

An anomaly may last for multiple consecutive time-series intervals and thus may trigger multiple alarms. Therefore, the number of alarms can be larger than the number of detected anomalies. Unless noted differently, we consider the number of detected anomalies in the evaluation.

If multiple anomalies overlap, the attribution of an alarm to a single anomaly may leave room for interpretation. For example, a long lasting anomaly may be superimposed by another short anomaly, with an alarm occurring while both anomalies are ongoing. As a general rule, if there are multiple anomalies in an alarm interval, we assume that the anomaly with the closest start time is responsible for the alarm. If there are more than one such anomaly, we prefer a relevant anomaly to an irrelevant anomaly.

In order to avoid confusion, we deliberately do not talk about true alarms and false alarms as it is often done in the literature (e.g., [BSM09]). In the context of change detection methods, an alarm is a false alarm if the statistical properties of the monitored variable have not changed. As we do not examine the output of a stochastic process with well known properties, we cannot distinguish between true and false alarms. Moreover, the classification in true and false alarms is unrelated to the distinction of relevant and irrelevant alarms according to this definition.

7.6 Implementation Details

The import of the flow records from the *flow-tools* files into the *MySQL* database is realized with a small Python script [Pyt10]. In the database, flow records are saved as rows in tables covering 30 minutes of flow data each. This partitioning provides a rough temporal classification of the flow records without using a database index. Further, it simplifies copy, move, and delete operations on parts of the data.

Another Python script has been implemented for generating measurement time series for the eight traffic metrics introduced in Section 7.3, namely the numbers of bytes, packets, flow records, distinct source and destination IP addresses, distinct source and destination port numbers, and the average flow duration. All these metrics can be easily obtained with SQL (Structured Query Language) queries. Filters can be optionally added with SQL WHERE clauses. Hence, the generation of the time series requires a single SQL query per table. The result is saved in a text file.

GNU Octave [GNU10], an open-source *Matlab* clone, turned out to be a suitable environment for analyzing the statistical properties of time series. *Octave* also enables us to implement and evaluate various residual generation and change detection methods with low programming effort. For our evaluation, we have implemented the Shewhart, CUSUM, and EWMA control charts as detection methods. In addition, we have programmed various residual generation methods, such as exponential smoothing and Holt-Winters forecasting, and principal component analysis. With the implemented *Octave* functions, we are able to combine different residual generation and change detection methods and to apply them with configurable parameter settings to traffic measurement time series.

Storing flow records in a database facilitates the root cause analysis thanks to the capabilities of the query languages, such as SQL. Root cause analysis aims at the identification of those flows that have contributed the most to a detected traffic anomaly. For example, database queries allow us to group the flow data by the source IP addresses in order to determine hosts that emit large amounts of packets or bytes, or that send packets to a large number of distinct destination IP addresses. By applying additional filters, we can narrow down the group of responsible flows and determine the corresponding destination IP addresses or port numbers. Knowledge about the responsible flows helps us to explain the cause of the detected anomaly, such as the start of a large data transfer between to hosts or scanning activity.

In Chapter 10, we present various algorithms which automate the identification of frequent anomaly causes. The algorithms rely on short sequences of SQL queries and have been implemented in Python scripts as well.

8. EVALUATION OF RESIDUAL GENERATION AND CHANGE DETECTION METHODS

8.1 Introduction

In this chapter, we evaluate the applicability of the modeling and residual generation methods presented in Chapters 4 and 5, as well as the detection of anomalies using control charts as introduced in Chapter 3. For this purpose, we consider the dataset presented in Section 7.2, which consists of flow records collected in an ISP backbone network. The flow records are converted into time series of the following eight metrics using the approach described in Section 7.3:

- number of bytes, packets, and flow records,
- number of distinct source and destination IP addresses,
- number of distinct source and destination port numbers,
- average flow duration.

In this chapter, the analyzed measurement time series are obtained from all flows and thus represent the entire IP traffic.

In Section 8.2, we summarize the statistical properties of the time series. Section 8.3 focuses on residual generation and change detection based on single metrics. We apply different time-series analysis methods and compare their capabilities to eliminate systematic changes and serial correlation. Furthermore, we compare three different control charts and evaluate their appropriateness for detecting anomalies in the residual time series of the number of bytes. Thereafter, we apply the best setup to the time series of all eight metrics and discuss which kinds of anomalies can be detected.

Section 8.4 is dedicated to multi-metric residual generation using PCA. We use batch-mode PCA based on non-robust and robust covariance estimates and apply control charts to time series of the T^2 and T_H^2 statistics as well as to individual y-scores. We compare the results to those obtained with incremental PCA using EWMA and EWMC estimators for the mean and covariances.

In Section 8.5, we summarize the results and draw conclusions how to deploy the considered anomaly detection methods in the most beneficial way. As will be seen, the main issue is to achieve a large proportion of relevant alarms that are of interest for the network administrator.

Tab. 8.1: Sample correlation matrix

Bytes	Packets	Records	Src addr	Dst addr	Src port	Dst port	Avg dur
1.0000	0.9838	0.8499	0.6858	0.3863	0.7795	0.7851	−0.5987
0.9838	1.0000	0.8811	0.7363	0.4162	0.8206	0.8267	−0.5901
0.8499	0.8811	1.0000	0.7603	0.5037	0.9317	0.9410	−0.7382
0.6858	0.7363	0.7603	1.0000	0.6509	0.8334	0.7970	−0.3729
0.3863	0.4162	0.5037	0.6509	1.0000	0.6621	0.4774	−0.3281
0.7795	0.8206	0.9317	0.8334	0.6621	1.0000	0.9431	−0.6614
0.7851	0.8267	0.9410	0.7970	0.4774	0.9431	1.0000	−0.6486
−0.5987	−0.5901	−0.7382	−0.3729	−0.3281	−0.6614	−0.6486	1.0000

8.2 Properties of Traffic Measurement Time Series

Figure 8.1 depicts the time series of the eight metrics. In all metrics, we observe seasonal variation which depends on the time of day. For most metrics, time-series values are lower at nighttime than at daytime. Only the average flow duration shows a reverse pattern with lower values during the day than during the night. In addition to the daily effect, there is a less pronounced weekly cycle with less traffic on Saturdays and Sundays. On October 3, which is a public holiday in Germany, we can recognize a slight decrease compared to the other weekdays as well.

In some intervals, there are isolated peaks in one or more metrics, indicating traffic anomalies. Most of the time, a peak in one metric coincides with a peak in another metric. For example, peaks in the number of bytes and packets often appear in the same time intervals, which suggests that these two metrics are highly correlated. However, none of the anomalies causes a peak in all eight metrics simultaneously. None of the metrics exposes the entire set of anomalies, either. Hence, it seems that different metrics must be taken into account to detect all kinds of anomalies.

The non-stationarity caused by the seasonal variation reappears in the sample autocorrelation functions displayed in Figure 8.2. It oscillates with a period of one day. We also see that the amplitude slightly increases at a lag of seven days which corresponds to one week. Except for the number of distinct destination IP addresses, the shape of the autocorrelation plots is very similar for the different metrics. The comparatively irregular shape of the destination IP addresses' sample autocorrelation very probably results from the long period of anomalous values around November 11, as can be seen in Figure 8.1. The time series of bytes, packets, and flows show strong and periodical seasonality which barely decays with increasing lag. For the number of distinct source and destination IP addresses, the sample autocorrelation is positive most of the time, which means that there is significant serial correlation in addition to the seasonality.

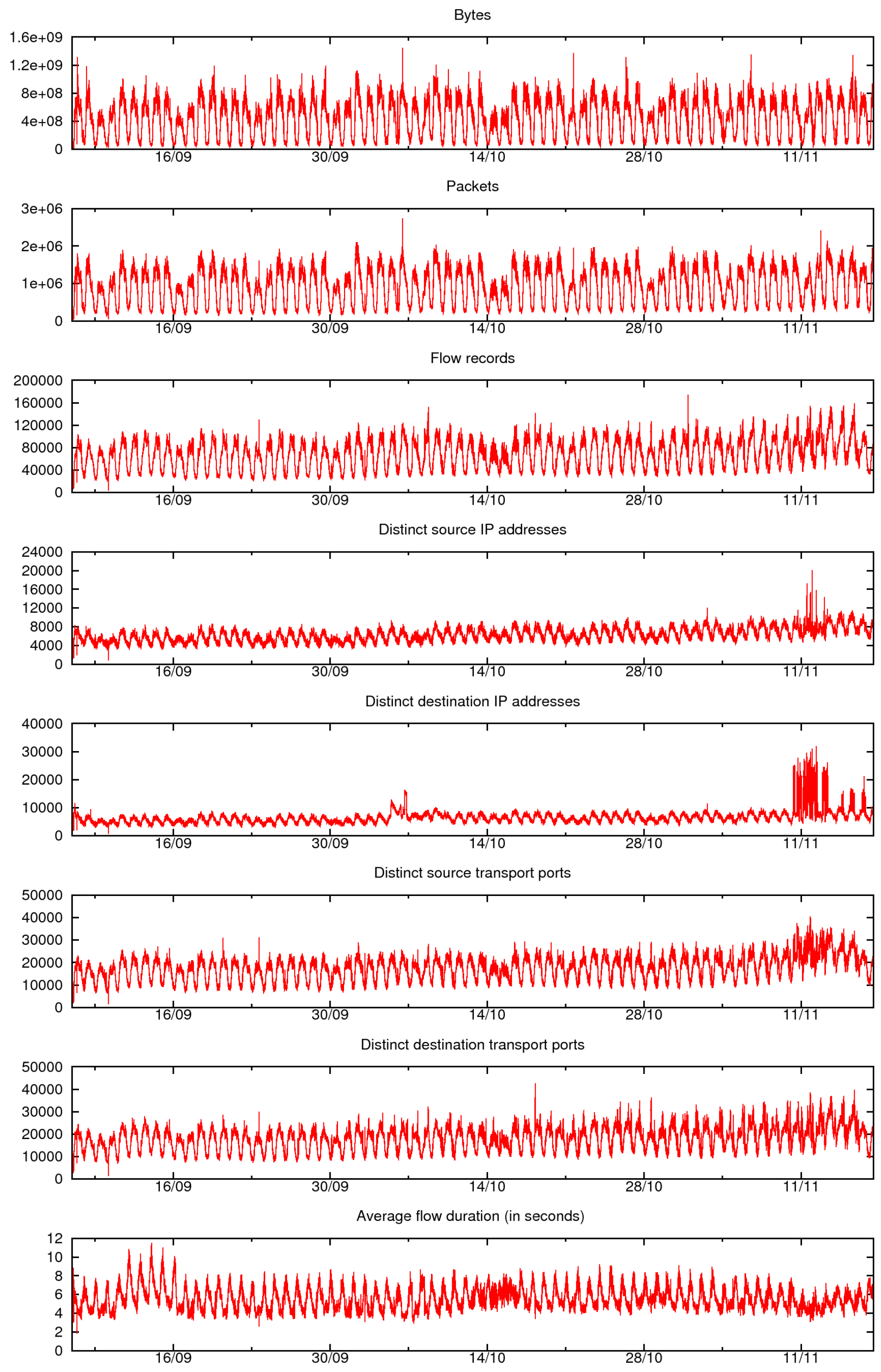


Fig. 8.1: Time series of all IP traffic

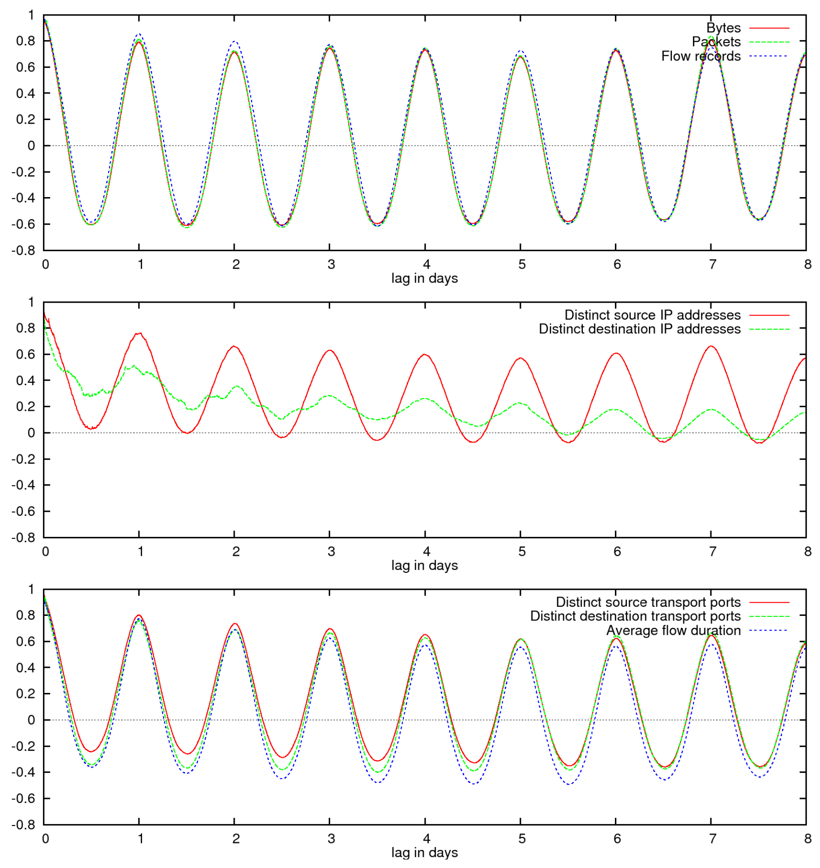


Fig. 8.2: Sample autocorrelation of all IP traffic

Tab. 8.2: Robust correlation matrix

Bytes	Packets	Records	Src addr	Dst addr	Src port	Dst port	Avg dur
1.0000	0.9920	0.9290	0.7682	0.7725	0.9109	0.8997	−0.7156
0.9920	1.0000	0.9410	0.8021	0.7992	0.9286	0.9173	−0.7001
0.9290	0.9410	1.0000	0.7973	0.7904	0.9694	0.9650	−0.7888
0.7682	0.8021	0.7973	1.0000	0.9418	0.8561	0.8399	−0.4253
0.7725	0.7992	0.7904	0.9418	1.0000	0.8380	0.8121	−0.4554
0.9109	0.9286	0.9694	0.8561	0.8380	1.0000	0.9800	−0.7351
0.8997	0.9173	0.9650	0.8399	0.8121	0.9800	1.0000	−0.7217
−0.7156	−0.7001	−0.7888	−0.4253	−0.4554	−0.7351	−0.7217	1.0000

Table 8.1 displays the sample correlation matrix of the metrics. As can be seen, the numbers of bytes and packets are highly correlated. There is also a high correlation between these two metrics and the number of flow records. We also observe a strong correlation between the numbers of distinct source and destination ports. This is not surprising because TCP and UDP traffic is usually bidirectional, thus we observe the same numbers as source and destination ports under normal conditions. However, a glance at the flow records reveals that several flows are unidirectional, which likely goes back to asymmetric routing, resulting in only one direction of the traffic being monitored by the backbone router.

The numbers of distinct source and destination ports are also correlated to the number of flow records. This phenomenon can be explained by the fact that client port numbers are ephemeral. For every new connection, a new client port is typically chosen from the range 49152 to 65535. The number of duplicate source ports is small as long as the number of parallel TCP connection and UDP streams is not too large. Indeed, the number of distinct source ports roughly equals half the number of flow records at nighttime when the traffic volume is low (see Figure 8.1).

Remarkably, the correlation between the numbers of distinct source and destination IP addresses is lower than the correlation of any of these metrics with the number of distinct source ports. The dynamic port assignment is the reason for dependencies between the numbers of distinct client port numbers and distinct client IP addresses. The rather low correlation between source and destination IP addresses is likely caused by the anomaly in November which has a much larger impact on the destination addresses than on the source addresses.

The average flow duration is negatively correlated with the other metrics. This opposed seasonal variation was also observed in Figure 8.1. It seems that an increase of the overall traffic volume during the day leads to a decrease of the average flow duration, probably because of increased web traffic with many short TCP connections.

For comparison, Table 8.2 shows a robust estimation of the correlation matrix based on the M-estimator with Cauchy-distributed weights presented in Section 5.3.2. All the non-diagonal elements are larger in magnitude compared to the sample correlation matrix. Having a look at the corresponding time-series plot in Figure 8.1, the reason becomes clear: extremely high values occurring between November 11 and November 14 distort the sample covariance whereas the robust estimation is less affected. This also explains why we now obtain a much higher correlation between the numbers of source and destination IP addresses.

Table 8.3 lists the estimated mean and standard deviation for all metrics. The robust estimates of the mean, resulting from the M-estimator, significantly differ from the non-robust sample means. Regarding the standard deviation, the difference is large for the number of destination IP ad-

Tab. 8.3: Estimated means and standard deviations

Metric	Mean		Standard deviation	
	non-robust	robust	non-robust	robust
Bytes	$4.5012 \cdot 10^8$	$3.8147 \cdot 10^8$	$2.6127 \cdot 10^8$	$2.5874 \cdot 10^8$
Packets	$9.2803 \cdot 10^5$	$7.9939 \cdot 10^5$	$4.7596 \cdot 10^5$	$4.7253 \cdot 10^5$
Flow records	$6.8823 \cdot 10^4$	$6.1214 \cdot 10^4$	$2.6063 \cdot 10^4$	$2.5663 \cdot 10^4$
Source IP addresses	$6.1239 \cdot 10^3$	$5.7203 \cdot 10^3$	$1.4398 \cdot 10^3$	$1.3049 \cdot 10^3$
Destination IP addresses	$6.5815 \cdot 10^3$	$5.7932 \cdot 10^3$	$2.6142 \cdot 10^3$	$1.3159 \cdot 10^3$
Source ports	$1.7596 \cdot 10^4$	$1.5797 \cdot 10^4$	$5.3262 \cdot 10^3$	$5.0154 \cdot 10^3$
Destination ports	$1.8171 \cdot 10^4$	$1.6529 \cdot 10^4$	$5.2840 \cdot 10^3$	$5.0700 \cdot 10^3$
Average flow duration	5.5596	5.7617	1.2208	1.2124

dresses, which again is the effect of the anomaly in November. Due to the non-stationary behavior of the measurement variables, these estimates are inadequate to characterize the time series. Yet, they give us an impression of the monitored traffic. For example, an average of 450.120.000 byte per interval corresponds to a rate of 1.5 megabyte per second, 928.030 packets per interval to 3093 packets per second. During the day, these values are much higher.

8.3 Single-Metric Residual Generation and Change Detection

This section deals with residual generation and detection of changes in the time series of a single metric. The residual generation is based on univariate time-series analysis methods, which means that correlation between different metrics is not taken into account.

8.3.1 Residual Generation Using Time-Series Analysis

Section 8.2 has shown that the measurement time series do not resemble the output of a stationary random process, mainly because of seasonal effects which depend on the time of day. As discussed in Section 4.3, the first time-series analysis step consists in the elimination of such systematic changes. For this purpose, we evaluated the following deseasoning techniques with respect to their capability to eliminate the effect of systematic changes from the measurement time series: seasonal differencing, additive and multiplicative deseasoning, exponential smoothing, and Holt-Winters forecasting.

Seasonal Differencing

As described in Section 4.3, seasonal differencing can be used to transform ‘integrated’ time-series models, such as ARIMA, into stationary (nonintegrated) models. Residuals of seasonal differencing are determined as differences of time-series values with a lag that corresponds to the seasonal period.

We applied differencing at lags of 288 and 2016 to the measurement time series, which means that we took the difference of time-series values which are one day or one week apart. As a result, seasonal variation was reduced, yet the correlation between residual values at a lag of one seasonal period remained very high. This is because the original time series are not free of anomalies. An anomalous value at time t_0 affects two values in the residual time series of the seasonal differencing, namely those at t_0 and $t_0 + s$, where s is the seasonal period. These additional peaks may lead to false alarms in the anomaly detection.

Deseasoning

Deseasoning is based on baseline and seasonal components which are estimated offline from training data. The residual time series describe the deviation from the estimated components. Deseasoning is called 'additive' or 'multiplicative' depending on whether the seasonal components are added to or multiplied by the baseline.

We estimated the baseline as well as additive or multiplicative seasonal components with periods of 288 and 2016 time intervals. The estimation was based on the first two weeks of measurement data. The residuals were then calculated for the remaining time-series values. Although the results were better than those of seasonal differencing, neither additive nor multiplicative deseasoning produced good residuals. We substituted the sample mean by the median, which is a more robust estimator of the mean, in order to reduce the effect of anomalous time-series values on the estimated components. However, this change did not improve the results. Hence, it seems that modeling seasonality alone is not sufficient because local variation in the time series may cause large deviations from the model.

Exponential Smoothing and Holt-Winters forecasting

The motivation for using exponential smoothing and Holt-Winters forecasting is not to predict future time-series values but to obtain the one-step-ahead prediction errors as residual time series. As described in Section 4.3.2, exponential smoothing and Holt-Winters rely on exponentially weighted moving averages estimating the model components. At the beginning, we initialize the components using the first values of the measurement time-series x_t:

- Exponential smoothing: $\hat{x}_2 = x_1$
- Holt-Winters: $L_s = 0$, $T_s = 0$, and $I_i = x_i$ for $i = 1, \dots, s$

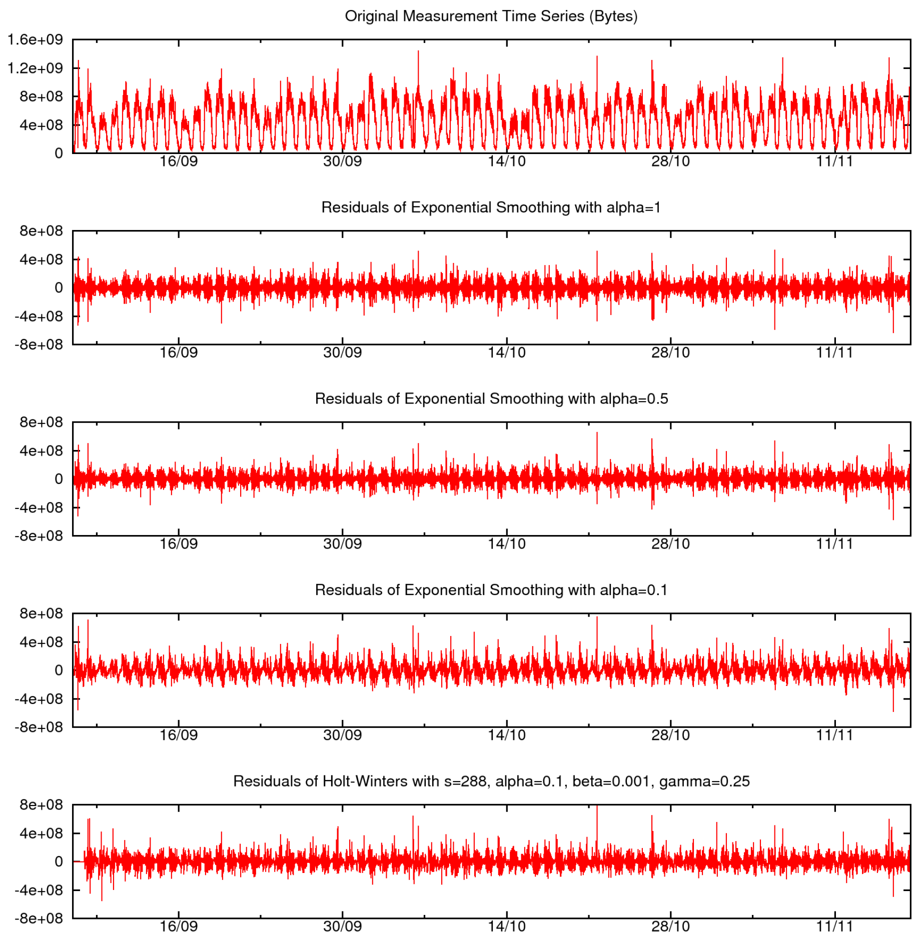


Fig. 8.3: Residuals of exponential smoothing and Holt-Winters forecasting

L_t, T_t, and I_t are Holt-Winters' mean, trend, and seasonal components; s is the seasonal period. The residual time series of exponential smoothing starts at the second time interval: $\epsilon_2 = x_2 - \hat{x_2} = x_2 - x_1$. Since the initialization of Holt-Winters requires s values, the residual time series starts at interval $t = s + 1$.

We investigated the residuals of exponential smoothing and Holt-Winters for different values of the smoothing constants. Figure 8.3 shows the byte count residuals for three different configurations of exponential smoothing ($\alpha = 1$, $\alpha = 0.5$, and $\alpha = 0.1$) and one setup of Holt-Winters forecasting with additive seasonal component ($s = 288$, $\alpha = 0.1$, $\beta = 0.001$, $\gamma = 0.25$). Exponential smoothing with $\alpha = 1$ is a special case because it takes the current value x_t as predictor for the next value x_{t+1}. This corresponds to differencing time-series values with lag 1.

Tab. 8.4: Mean square of residuals for number of bytes

Metric	ES($\alpha = 1$)	ES($\alpha = 0.8$)	**ES**($\alpha = 0.5$)	ES($\alpha = 0.1$)	HW
Bytes	$5.0 \cdot 10^{15}$	$4.3 \cdot 10^{15}$	$3.9 \cdot 10^{15}$	$8.0 \cdot 10^{15}$	$5.3 \cdot 10^{15}$
Packets	$1.3 \cdot 10^{10}$	$1.1 \cdot 10^{10}$	$9.3 \cdot 10^{9}$	$2.1 \cdot 10^{10}$	$1.2 \cdot 10^{10}$
Records	$3.8 \cdot 10^{7}$	$3.4 \cdot 10^{7}$	$3.1 \cdot 10^{7}$	$7.3 \cdot 10^{7}$	$5.4 \cdot 10^{7}$
Src addr	$3.2 \cdot 10^{5}$	$2.8 \cdot 10^{5}$	$2.5 \cdot 10^{5}$	$3.0 \cdot 10^{5}$	$3.0 \cdot 10^{5}$
Dst addr	$1.5 \cdot 10^{6}$	$1.3 \cdot 10^{6}$	$1.2 \cdot 10^{6}$	$1.6 \cdot 10^{6}$	$1.8 \cdot 10^{6}$
Src port	$2.4 \cdot 10^{6}$	$2.1 \cdot 10^{6}$	$1.9 \cdot 10^{6}$	$3.6 \cdot 10^{6}$	$3.1 \cdot 10^{6}$
Dst port	$3.4 \cdot 10^{6}$	$2.9 \cdot 10^{6}$	$2.5 \cdot 10^{6}$	$4.4 \cdot 10^{6}$	$4.1 \cdot 10^{6}$
Avg dur	$2.3 \cdot 10^{-1}$	$2.0 \cdot 10^{-1}$	$1.7 \cdot 10^{-1}$	$2.4 \cdot 10^{-1}$	$2.2 \cdot 10^{-1}$

Regarding the different settings for exponential smoothing, the seasonal variation of the mean is not visible any more except for $\alpha = 0.1$. $\alpha = 0.5$ provides the best results: Obvious anomalies in the original data appear as clear impulses whereas the variability during normal traffic is relatively low. We also studied exponential smoothing with $\alpha = 0.8$ and obtained larger background variability between peak values. This visual impression is confirmed by the mean square of the residuals, which corresponds to the mean squared prediction error. The values in Table 8.4 show that exponential smoothing with $\alpha = 0.5$ provides the best forecasting for all metrics.

Similar to our examinations of exponential smoothing, we tested various parameterizations of Holt-Winters forecasting with different smoothing constants and seasonal periods of one day and one week (i.e., $s = 288$ and $s = 2016$). However, the residual time series were quite similar to those obtained by exponential smoothing despite the additional complexity of Holt-Winters forecasting. The parameter setting used in Figure 8.3 effectively reduces the seasonal variation and exposes various anomalies in the measurement data. For larger α, Holt-Winters resembles more and more simple exponential smoothing since the fast update of the mean component absorbs the main part of variation. For smaller α, differences between workday and weekend become remarkable. The effect of the trend component is negligible because there is no remarkable trend in the measurements. Smaller values for γ inhibit that improper initialization values of the seasonal component are rapidly smoothed away. A seasonal period of 2016 instead of 288 did not improve the results.

The residual time series are obviously not stationary with respect to the variance because the variability during the day is much higher than at nighttime. Since a periodic pattern in the variance is a sign of multiplicative seasonal effects, we also tried Holt-Winters with multiplicative seasonal components. Yet, we did not achieve any improvement compared to additive components.

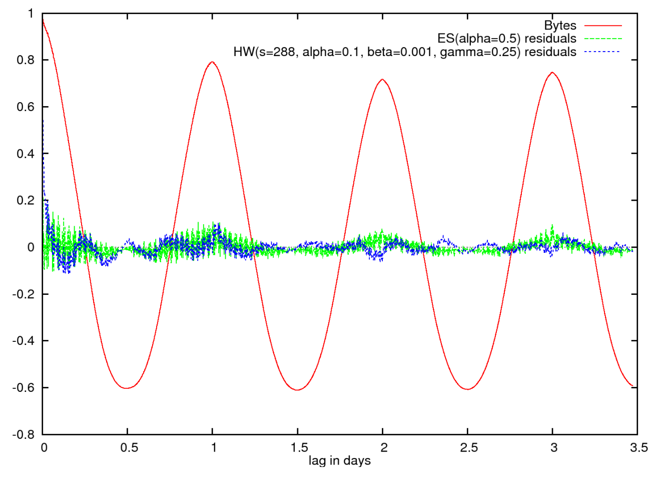


Fig. 8.4: Sample autocorrelation of the residuals

Regarding the mean square of the residuals, Holt-Winters forecasting did not yield lower values than exponential smoothing with $\alpha = 0.5$. As an example, Table 8.4 shows the values for the parameters used in Figure 8.3. The probable reason why Holt-Winters forecasting does not outperform exponential smoothing is the long seasonal period of 288 or 2016 intervals, respectively. The effect of the seasonal variation on consecutive values is small compared to the shifts of the baseline component. Furthermore, it is difficult to obtain good estimates of the seasonal components since we get only one value per day or week.

Figure 8.4 shows the sample autocorrelation of the original byte count time-series and the corresponding residual time series. As expected, the autocorrelation of the residuals attenuates quite quickly. We still observe oscillation, but with much smaller amplitude.

As a conclusion, we can state that exponential smoothing with $\alpha = 0.5$ enables the elimination of seasonal variation quite well for all considered metrics. The differences between the residuals of exponential smoothing and Holt-Winters forecasting are small without any advantage of the later. The increased complexity of the Holt-Winters does not improve the results.

8.3.2 Change Detection Using Shewhart, CUSUM, and EWMA Control Charts with Adaptive Control Limits

We have implemented three different change detection methods to detect changes in the residual time series: Shewhart control chart of individuals, CUSUM control chart, and EWMA control chart. In order to compare these three methods, we apply them to the byte count residual time series of exponential smoothing and Holt-Winters forecasting. Although the residuals have much more favorable statistical properties than the original measurement variables, they cannot be assumed to be stationary and free of serial correlation during normal traffic. Hence, we cannot calculate control limits for a predefined false positive rate since the necessary condition of independent and identically distributed observations is not fulfilled.

Instead, we vary the parameters of the detection methods to study their effect on the detection result empirically. The cause of every detected anomaly is identified by searching the original flow data for those flows which are most likely responsible for the traffic change.

As the statistical properties of the residual time series are unknown, we build the control charts on adaptive control limits based on the moving estimation of the residuals' standard deviation $\hat{\sigma}$. A selection of such estimators has been presented in Section 4.5. Unless otherwise noted, we used the exponentially weighted mean square (EWMS) as estimator for $\hat{\sigma}^2$, assuming that the residual time series has zero mean ($\mu = 0$).

Shewhart Control Chart

Figure 8.5 shows the original time series of the number of bytes on top and the Shewhart control charts of the residual time series of exponential smoothing ($\alpha = 0.5$) and Holt-Winters ($s = 288$, $\alpha = 0.1$, $\beta = 0.001$, $\gamma = 0.25$) below. The adaptive control limits at $\pm 6\hat{\sigma}$ are depicted as green lines in the diagrams; the centerline is zero. The smoothing constant of the EWMS estimator for $\hat{\sigma}^2$ is set to $\rho = 0.01$. The dotted vertical lines mark alarm intervals in which the residual values fall beyond the control limits. The residuals of exponential smoothing breach the control limits fifteen times (two alarms are very close together on early October 6). Apart from four alarms, the identical set of anomalies is detected in the Holt-Winters residuals.

Table 8.5 shows the causes of the alarms which have been found by inspecting the original flow records. As can be seen, all of the anomalies go back to irregular traffic observed in the network. However, apart from the anomalous increase in SMTP traffic on October 6, which is certainly the result of a mailbomb or a flood of spam mails, the detected anomalies seem to be harmless. The first two occurrences of RSF-1 traffic, one of the HTTP anomalies, and the second increase of SMTP traffic on October 6 are detected in the residuals of exponential smoothing only. The remaining anomalies are found in both control charts. The RSF-1 traffic belongs to a high-availability and cluster middleware application by HAC (High-Availability.Com) [Hig10].

Although most of the detected alarms can be linked to anomalous looking time segments in the byte count time series (see Figure 8.5), some of the alarms do not coincide with an extremely high or low number of bytes. An example is the mailbomb which does not cause larger byte counts than observed at normal daytimes. This anomaly would not have been detected by

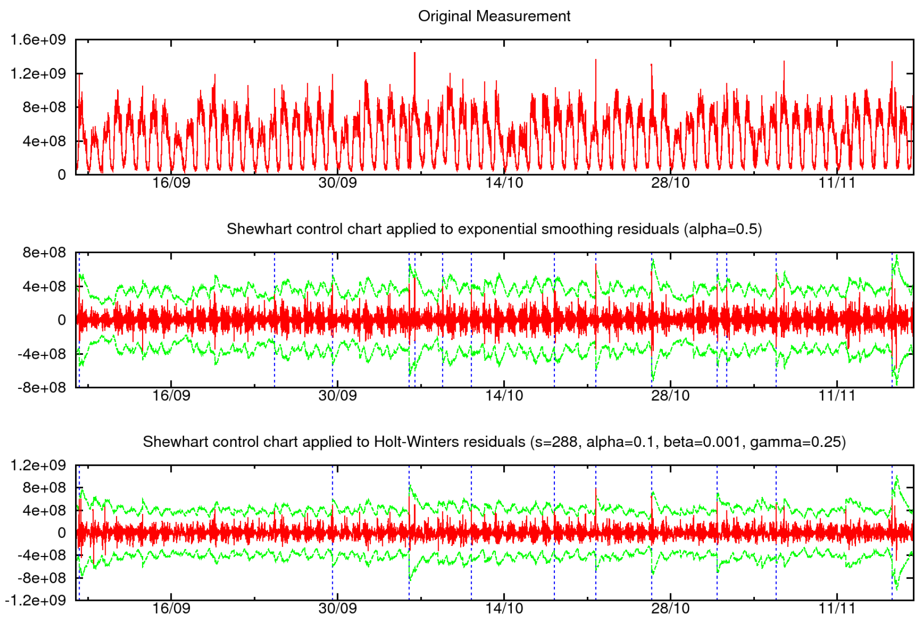


Fig. 8.5: Shewhart control chart with control limits $\pm 6\hat{\sigma}$ $(\rho = 0.01)$

a fixed threshold applied to the original measurement time series. However, as the mailbomb causes an abrupt and anomalous traffic increase during the night, it results in a peak value in the residuals exceeding the control limits of the control chart.

After decreasing the control limits to $\pm 5\hat{\sigma}$, we obtain 16 additional alarms for exponential smoothing and 19 additional alarms for Holt-Winters. Only three of these additional alarms are detected in both residual time series. Most alarms are caused by large HTTP downloads. Some of the alarms seem to be the result of regular traffic fluctuation since we have not been able to identify any specific causes. In summary, we can conclude that the anomalies in the number of bytes are mostly due to harmless traffic, except the nightly SMTP anomaly.

In order to understand the influence of the standard deviation estimation, we have calculated the Shewhart control limits with other estimators than EWMS (see Section 4.5) but have not found any notable differences between the estimates obtained by EWMS, exponentially weighted absolute deviation, and exponentially weighted square successive difference. The control limits based on exponentially weighted absolute successive difference deviate from the others and result in some alarms at time intervals where no obvious deviation from normal traffic can be observed. A probable reason is the remaining serial correlation in the residuals, which introduces bias to this estimator.

We have also tested different values of the smoothing constant ρ. For

Tab. 8.5: Shewhart detection results (limits = $\pm 6\hat{\sigma}$, $\rho = 0.01$)

Time	Cause
08/09 08:05	FTP download (approx. 640 MBytes, likely a CD image)
24/09 18:50	RSF-1 data transfer between two hosts on UDP port 1195
29/09 15:45	HTTP download (approx. 160 MBytes)
06/10 01:45	large amount of SMTP traffic (possibly mailbomb or spam) during three hours
06/10 02:00	peak of the SMTP anomaly that started 15 minutes before
06/10 13:15	large amount of SMTP traffic from one client to several servers (maybe another mailbomb or delivery of queued e-mails)
08/10 21:50	RSF-1 data transfer between two hosts on UDP port 1195
11/10 07:25	HTTP download (maximum of 355 MBytes in one interval)
18/10 07:15	HTTP download (maximum of 195 MBytes in one interval)
21/10 19:00	HTTP download (maximum of 524 MBytes in one interval)
26/10 11:05	FTP download
31/10 22:35	FTP upload, increased HTTP and DNS traffic
01/11 18:15	increased HTTP traffic
05/11 21:45	RSF-1 data transfer between two hosts on UDP port 1195
15/11 15:40	HTTP download

larger ρ, the control limits become highly varying with exposed daily periodicity. As a result, more and more alarms are triggered when the traffic volume increases in the morning hours. Hence, all in all, EWMS with $\rho = 0.01$ is a good estimator for adaptive control limits.

CUSUM Control Chart

Our implementation of the CUSUM control chart consists of two CUSUM decision functions to detect shifts in positive and negative direction (see Section 3.2.3). Reference value and threshold are configured as multiples of the standard deviation $\hat{\sigma}$ which is estimated by EWMS with smoothing constant $\rho = 0.01$ as in the Shewhart control chart

The charts in Figure 8.6 show the maximum of the two CUSUM statistics as well as the adaptive threshold for a reference value of $2\hat{\sigma}$ and a threshold of $5\hat{\sigma}$. With this setting, CUSUM triggers nine alarms for exponential smoothing residuals and 23 alarms for Holt-Winters residuals. Eight alarms triggered by the exponential smoothing residuals are also detected in the Shewhart control chart. In the case of the Holt-Winters residuals, the CUSUM control chart leads to a couple of new alarms, most of them related to HTTP and FTP traffic. For some alarms, we have not found any specific reason.

We have applied the CUSUM control chart with different parameters without finding any additional anomalies of interest. For small thresholds

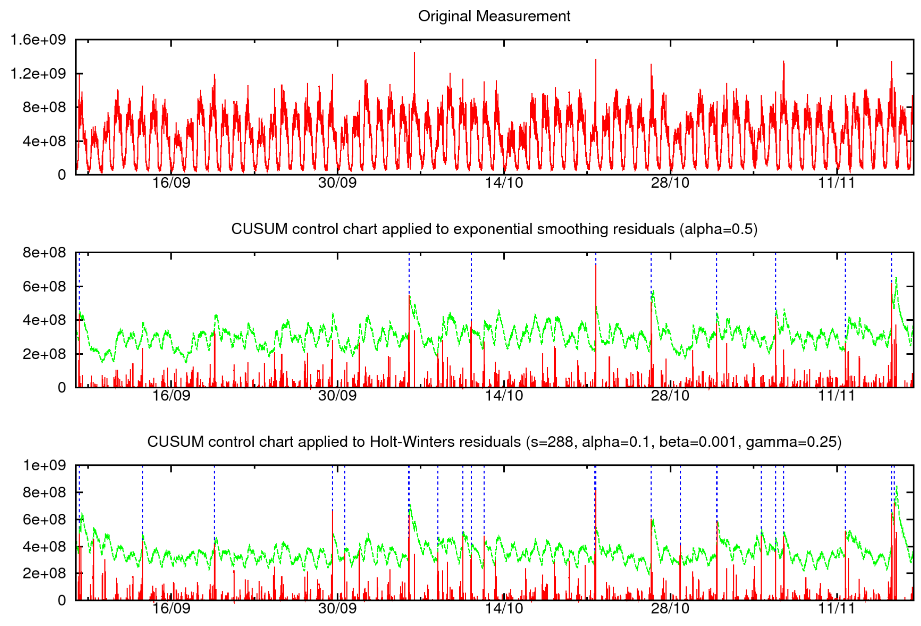


Fig. 8.6: CUSUM control chart with reference $2\hat{\sigma}$ and threshold $5\hat{\sigma}$

and reference values, CUSUM tends to raise many alarms which cannot be linked to anomalous traffic. Since traffic anomalies in the original time series typically produce short increases in the residual time series, CUSUM's capability to detect small sustained shifts of the mean is of limited utility. As a result, we do not benefit from CUSUM's methodological advantage over the Shewhart control chart. All in all, the CUSUM control chart does not provide better detection results than the Shewhart control chart for the given residual time series.

EWMA Control Chart

As a third change detection method, we have implemented the EWMA control chart which calculates the exponentially weighted moving average of the residuals and compares it against upper and lower control limits. The moving average has a smoothing effect which reduces the normal variation and helps detecting small but sustained shifts of the mean. Since traffic anomalies are mainly visible as short peaks in our residual time series, we expect a similar outcome as for the CUSUM control chart, which is that the properties of the EWMA control chart do not improve the detection results.

With decreasing smoothing constant, EWMA accentuates more and more the remaining seasonal variation which has not been entirely removed during the residual generation. Therefore, we apply the EWMA control chart with a rather large value of $\lambda = 0.25$. As explained in Section 3.3.3, the con-

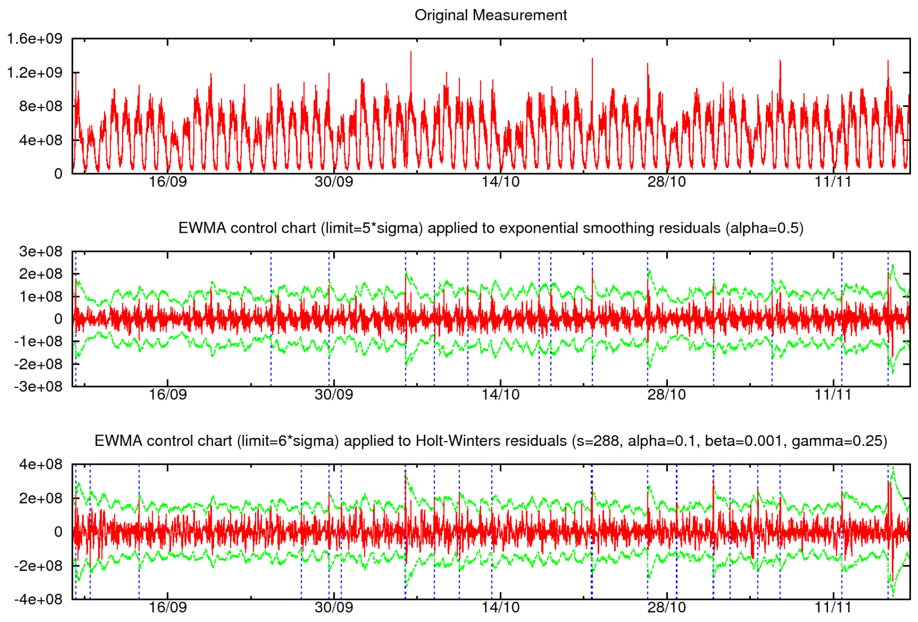


Fig. 8.7: EWMA control chart with $\lambda = 0.25$

trol limits are $\pm L\sqrt{\frac{\lambda}{2-\lambda}}\hat{\sigma}$, where $\hat{\sigma}$ is the EWMS estimator of the residuals' standard deviation, again calculated with smoothing constant $\rho = 0.01$.

Figure 8.7 depicts the EWMA control charts for the residuals of exponential smoothing and Holt-Winters with control levels $L = 5$ and $L = 6$, respectively. These levels result in a similar number of alarms for both residual time series. As expected, most of the alarms also appear in the Shewhart control chart. For narrower control limits, the EWMA control chart generates additional alarms which mainly go back to large data transfers.

Discussion

Our experimentations show that the CUSUM control chart is not a universal remedy for traffic anomaly detection problems despite its frequent utilization in the related work (see Section 3.3.5). In our case, the CUSUM control chart risks to trigger alarms when evident traffic anomalies are absent. The EWMA control charts works fine for large smoothing constants ($\lambda = 0.25$), yet without providing better results than the Shewhart control chart of individuals.

The theoretical advantage of CUSUM and EWMA control charts is their capability to detect small sustained shifts in the mean. However, we do not have such shifts in the time series of prediction errors since the forecasting methods adapt quite quickly to a change in the original measurement variables. Furthermore, serial correlation is not completely eliminated in the residual time series, which may lead to false alarms. Since the Shewhart control chart of individual considers individual observations, the decision is not affected by serial correlation.

We use adaptive control limits and thresholds which are based on moving estimators of the residuals' standard deviation. The benefit of this approach is that the detection automatically adapts to the variation in the residuals. In our implementation, the moving estimators are continuously updated, even after an anomaly has been detected. As a consequence, control limits and thresholds are temporarily increased above the normal level which may inhibit the detection of further anomalies that follow in a short distance. This problem could be avoided by suspending the update of the estimator in the case of an alarm, yet at the risk of not accounting for an abrupt increase of normal variation either.

There is no big difference between the alarms detected in the residuals of exponential smoothing and Holt-Winters forecasting. This confirms that simple exponential smoothing is an appropriate residual generation method for our purpose. Independently of the applied change detection methods, most of the anomalies detected in the number of bytes go back to harmless traffic variation. In the next section, we compare the detection results obtained for different metrics.

8.3.3 Comparison of Different Metrics

In the preceding subsection, we compared the change detection results of different residual generation methods and control charts considering the byte count metric. The Shewhart control chart of individuals applied to the residuals of exponential smoothing with $\alpha = 0.5$ provided good change detection results, despite of its simplicity. However, most of the detected anomalies could not be associated with any attack traffic, network failures, or other events of interest.

In this subsection, we apply the Shewhart control charts with different control limits to the exponential smoothing residuals of all eight metrics and identify the causes of the triggered alarms. The smoothing constant for estimating the residuals' standard deviation is $\rho = 0.01$ as before.

As described in Section 7.5, an anomaly is classified as relevant if it can be associated with a pattern of harmful or unwanted traffic or severe network problems by inspecting the original flow records. Examples of relevant anomaly causes are network and port scans, password guessing, and network outages. Irrelevant anomalies are related to large data transfers or traffic variations caused by regular services, such as DNS. Lacking additional context information, it is possible that we do not always find the actual cause of a traffic anomaly. Furthermore, our classification into relevant and irrelevant anomalies is subjective to a certain extend. For example, some of the alarms which we classify as irrelevant could still be of interest for the network administrator and vice versa. Nevertheless, our classification approach enables a qualitative comparison of the detection results obtained for different metrics.

Figure 8.8 shows the numbers of relevant and irrelevant anomalies detected in every metric in dependence of the control limits. If an anomaly lasts for several time intervals, the residual time series may cross the control limits multiple times, causing more than one alarm. Such anomalies are counted only once. In the given setup, there is no big difference between anomaly counts and alarm counts since exponential smoothing in combination with adaptive thresholds rarely leads to multiple alarms triggered by the same anomaly. The control limits decrease from $\pm 7\hat{\sigma}$ to $\pm 5\hat{\sigma}$. As expected, the numbers of relevant and irrelevant anomalies increase with decreasing control limits. In those cases where limits at $\pm 5\hat{\sigma}$ yield more relevant than irrelevant anomalies, we also evaluate $\pm 4\hat{\sigma}$ to depict the further trend.

We are interested in metrics which allow detecting a large number of relevant anomalies while the number of irrelevant anomalies is small. As can be seen in Figure 8.8, byte and packet counts do not fulfill this criterion since most of the anomalies are classified irrelevant regardless of the control limits. On the other hand, the majority of the anomalies found in the destination IP addresses and source ports are relevant for the considered range of control limits. The remaining metrics yield more relevant anomalies than irrelevant anomalies as long as the control limits do not fall below a certain level.

Now, we have a closer look at the anomalies detected with adaptive thresholds at $\pm 6\hat{\sigma}$. In this case, the largest number of anomalies is triggered by the byte count time series (14), yet only 2 are classified as relevant (14%). The largest proportion of relevant anomalies is found in the number of distinct source ports (100%), destination IP addresses (88%), and source IP addresses (75%). The identified causes of the relevant anomalies are listed in Table 8.6. The bit vector in the last column indicates by 1s in which metrics each anomaly is detected, using the same ordering as before: bytes, packets, records, source addresses, destination addresses, source ports, destination ports, average duration. In the following, we summarize and interpret the results for every metric:

Byte count: This metric causes the largest number of anomalies. Most of them go back to individual large data transfers (HTTP, FTP, RSF-1). There is only one event of interest: the enormous increases of SMTP traffic on October 6, likely caused by a mailbomb or spam. This anomaly does not cause any significant changes in any other metric. Therefore, we would risk to miss this alarm if we exempted the byte count time series from change detection.

Packet count: Six anomalies are found in this metric. As for the byte count, most of the anomalies are caused by large data transfers. The two anomalies classified as relevant are also detected in three other

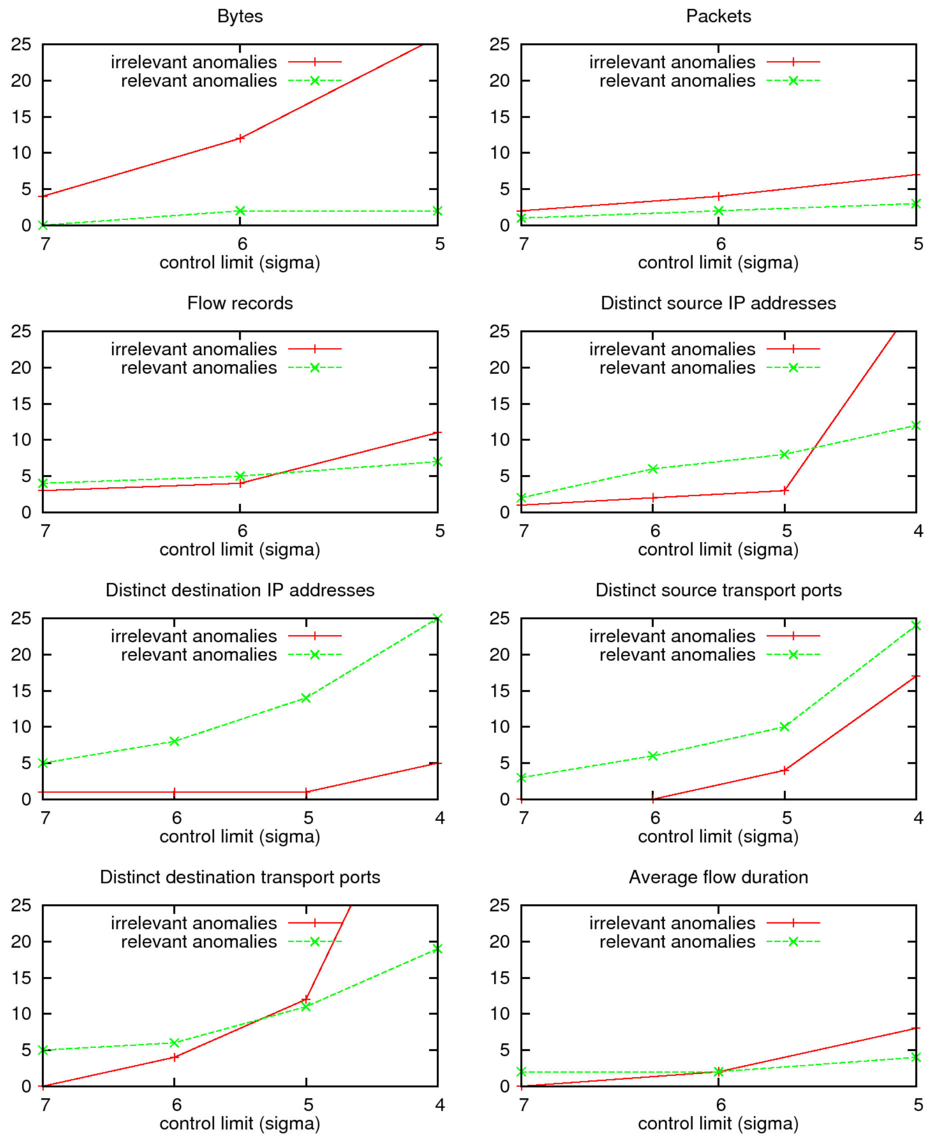


Fig. 8.8: Alarms of Shewhart control chart with adaptive control limits applied to ES residuals

Tab. 8.6: Relevant anomalies detected by Shewhart control chart with adaptive control limits applied to ES residuals

Time	Cause	Metrics
08/09 17:00 - 08/09 17:10	SMB network scan	00001000
10/09 02:25 - 10/09 02:25	SSH password guessing	00000101
10/09 07:15 - 10/09 07:15	Network failure	01100110
14/09 17:00 - 14/09 17:25	SSH password guessing	00000100
17/09 00:40 - 17/09 00:50	SSH password guessing	00000010
23/09 17:50 - 23/09 18:15	SSH password guessing	01100110
26/09 01:20 - 26/09 01:50	SSH password guessing	00000110
03/10 04:20 - 03/10 05:40	SSH password guessing	00100111
06/10 01:45 - 06/10 04:00	SMTP (mailbomb)	10000000
06/10 10:10 - 06/10 10:10	End of an SMB network scan	00001000
06/10 13:15 - 06/10 13:20	SMTP (mailbomb)	10000000
06/10 17:00 - 07/10 12:25	SMB network scan	00001001
18/10 08:20 - 18/10 08:45	Port scan	00100010
18/10 08:50 - 18/10 08:50	End of port scan	00000010
05/11 14:55 - 05/11 15:00	Port scan	00100000
10/11 08:05 - 10/11 08:30	HTTPS network scan	00001000
10/11 08:50 - 10/11 09:10	HTTPS network scan (cont.)	00001000
10/11 18:10 - 11/11 00:50	HTTPS network scan (cont.)	00010000
11/11 12:35 - 11/11 14:25	HTTPS network scan (cont.)	00010000
11/11 20:20 - 11/11 21:20	HTTPS network scan (cont.)	00010000
11/11 22:05 - 12/11 01:45	HTTPS network scan (cont.)	00010000
12/11 09:00 - 12/11 10:00	HTTPS network scan (cont.)	00010000
12/11 22:40 - 13/11 06:35	HTTPS network scan (cont.)	00010000
14/11 16:15 - 14/11 17:10	ICMP network scan	00001000
15/11 10:00 - 15/11 10:50	ICMP network scan	00001000
16/11 11:05 - 16/11 11:50	ICMP network scan	00001000

metrics, namely in the number of records, source, and destination ports. Thus, we can exempt the packet count time series from change detection without losing any relevant anomalies.

Record count: Nine anomalies are detected in the time series of this metric. Five of them are classified as relevant and go back to a network failure, port scans, and large numbers of failed SSH login attempts. The failed login attempts are very probably caused by brute-force password guessing. Four relevant anomalies are also detected in other metrics among which we always find the number of distinct destination ports. The remaining port scan is not detected in any other metric, even after reducing the control limits to $\pm 4\hat{\sigma}$. The reason is that each of the 5,400 concerned ports is scanned multiple times, resulting in a much larger increase in the number of records than in the number of distinct destination ports. The irrelevant alarms are caused by HTTP traffic.

Distinct source addresses: Six out of eight anomalies are classified as relevant because they can be linked to the long-lasting HTTPS network scanning activities of a single host. A large number of source addresses is not a usual sign for a network scan. In the alarm intervals, however, the number of flows returned from the scanned hosts is increased. For the irrelevant alarms, we could not find any other reason than an increased number of flows caused by DNS traffic. Since the relevant anomalies are detected in other metrics as well, this metric is dispensable.

Distinct destination addresses: We get eight relevant anomalies which all go back to TCP and ICMP network scanning activities. The only irrelevant alarm is caused by increased DNS traffic. Hence, the number of distinct destination addresses is an important and reliable metric to detect scans.

Distinct source ports: We obtain a ratio of 100 percent relevant alarms for this metric. One of the anomalies is caused by a network failure, the others are provoked by SSH password guessing. Thus, this metric is very useful to detect password guessing.

Distinct destination ports: Six out of ten anomalies are classified as relevant. As causes, we identify SSH password guessing, a port scan, and a network failure. The irrelevant alarms go back to DNS anomalies.

Average flow duration: For this metric, we get only four alarms with different causes: an HTTP anomaly, two SSH password guessing attempts, and an SMB network scan. Since the relevant anomalies are detected in other metric as well, this metric can be omitted from the anomaly detection without missing any important events.

In summary, the numbers of bytes and packets turn out to be the metrics with the least interesting detection results. It is much more interesting to detect changes in the number of source and destination addresses and ports. Just like the number of records, these metrics allow detecting different kinds of scans and SSH password guessing, yet with a much smaller number of irrelevant alarms. Monitoring the average flow duration is not very useful, at least if the entire IP traffic is considered. Since regular variation in DNS traffic is responsible for multiple irrelevant anomalies, filtering out traffic from and to UDP port 53 (which is the well-known port for DNS) would very likely decrease the number of irrelevant anomalies, in particular in the case of the cardinality metrics.

8.3.4 Constant Control Limits

Up to now, we have used control charts with adaptive control limits based on moving estimators of the standard deviation. The advantage of this approach is that no a-priori knowledge about the statistical properties of the measurement variables is required. The objective of this subsection is to evaluate how the detection results change if control charts with constant control limits are deployed. For this purpose, we use the first two weeks of the measurement time series as training data and the remaining observations for anomaly detection

From the training data, we estimate the mean and standard deviation of each measurement variable. With these estimates, we standardize the measurement time-series in order to obtain values that are more or less in the same value range. Next, we apply exponential smoothing with $\alpha = 0.5$ for residual generation. The resulting residual time series serve as input to the Shewhart control chart of individuals with constant control limits.

Figure 8.9 displays the anomaly counts in dependence of the control limits varied from ± 3 to ± 1.6. In the case of the source and destination IP addresses, the numbers of alarms are significantly larger than the displayed numbers of anomalies. The reason is that many repeated alarms are triggered during the long-lasting HTTPS network scan in November which proceeds with varying intensity. We also observe that alarms are often triggered at the beginning of an anomaly and after it has ended because a peak occurs in both intervals of the residual time series. In this case, both alarms are accounted as relevant anomalies, which results in slightly higher numbers of anomalies compared to the control charts with adaptive control limits where the second peak is usually not detected due to increased control limits.

The numbers of relevant and irrelevant anomalies detected in the destination IP addresses do not increase monotonically with decreasing control

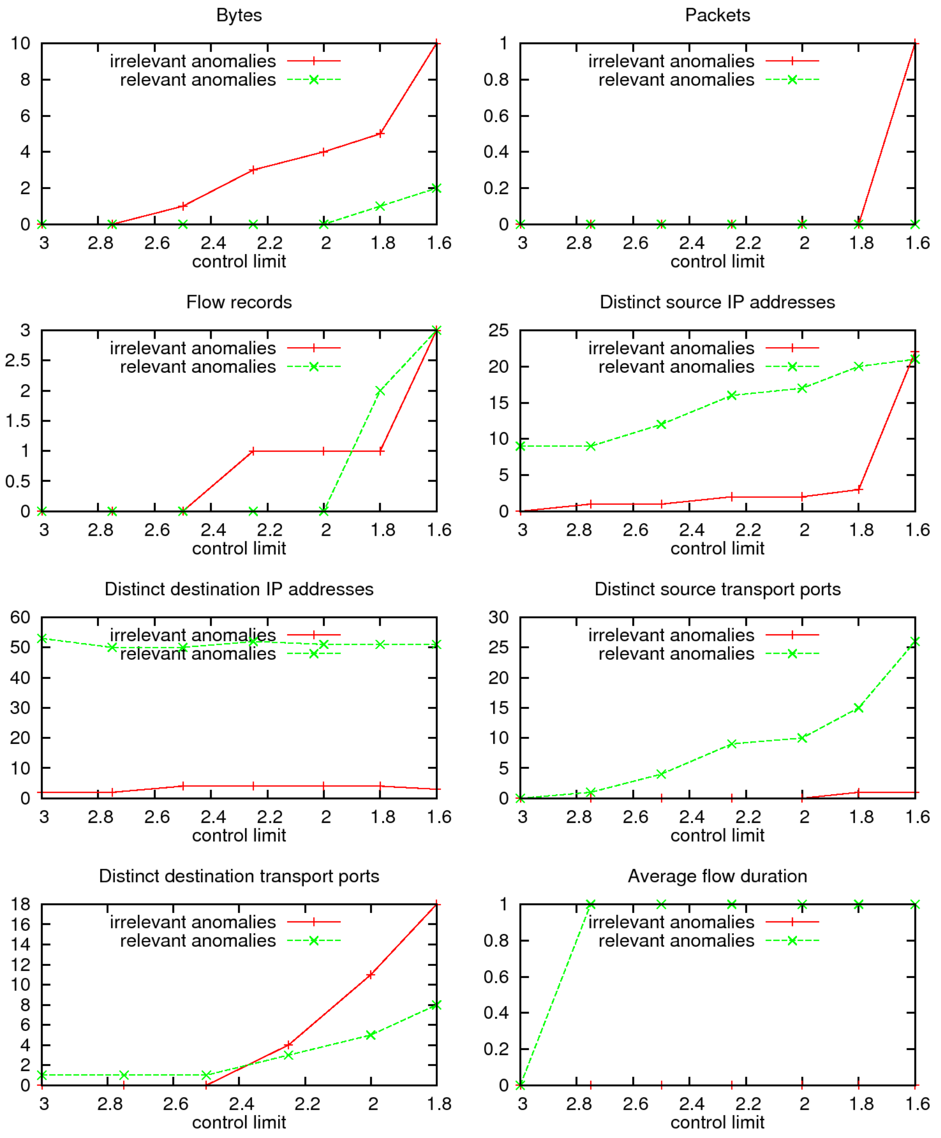


Fig. 8.9: Alarms of Shewhart control chart with constant control limits applied to ES residuals

limits. The reason is that we only account those time intervals as alarms where the residual value crosses one of the control limits. If two events overlap or occur directly one after the other, it may happen that the residual value stays above the control limits and thus does not trigger a new alarm.

A comparison of the different metrics leads to similar conclusions as the control charts with adaptive control limits. The numbers of distinct destination IP addresses and source ports are the most interesting metrics, mainly due to their sensitivity to network scans. The proportion of relevant anomalies is also very high for the number of distinct source IP addresses. However, nearly all of them are related to different phases of the HTTPS network scan. With adaptive control limits, we detected several password guessing attacks in the same metric, which we do not find now. In total, we detect a few additional relevant anomalies which have not been found with adaptive control limits, while some previously detected ones are missing.

Setting appropriate values for the constant control limits is not easy. Even though we have standardized the original measurement variables, the different metrics trigger very different numbers of alarms. The reason is that the applied standardization assumes normality and stationarity. Control charts with adaptive control limits are much easier to use as they neither require as much tuning of the control limits nor any training data to estimate the standard deviation.

8.4 Multi-Metric Residual Generation and Change Detection

In this section, we deploy PCA techniques for multi-metric residual generation and change detection. In contrast to the single-metric approach, multiple metrics are jointly taken into consideration, which allows detecting changes in the correlation structure of the measurement variables.

As explained in Chapter 5, PCA is tightly coupled with the covariance matrix. In our case, the original variables are measured in different units. Therefore, we apply PCA to standardized measurement time series, which means that we first subtract the average and divide the result by the estimated standard deviation. This approach has been described in Section 5.3.1, where we have also mentioned that the covariance matrix of the standardized time series is equal to the correlation matrix of the original variables.

In a first step, we apply conventional batch-mode PCA to the measurement time series. Thereby, we compare the results obtained with the sample mean and correlation matrix to those resulting from the M-estimator mean and correlation matrix. The later estimates are more robust against outliers in the training data. Secondly, we deploy incremental PCA which is based on the EWMC estimator.

8.4.1 Batch-Mode PCA

Batch-mode PCA requires training data to calculate the PCs. For our evaluation, we use the first two weeks of the measurement time series for training and the remaining data for change detection.

As mentioned in Section 5.3.2, the conventional sample covariance and correlation matrices are susceptible to outliers. Depending on the frequency and intensity of outliers and anomalies in the training data, the PCs deviate from the correlation structure of normal traffic. Therefore, we also use robust estimates of the M-estimator family with Cauchy-distributed weights and compare the PCA results to those obtained with the sample mean and correlation matrix.

In the following, we discuss the differences between the two estimation approaches. Then, we evaluate the detection results obtained with Hotelling's T^2 statistic and Hawkins' T^2_H statistic. Thereafter, we apply the Shewhart control chart to the time series of individual y-scores and examine the triggered alarms. Finally, we combine batch-mode PCA and exponential smoothing in the residual generation process and evaluate if this approach has any benefits.

Comparison of Sample and M-estimator Mean and Correlation Matrix

Mean, standard deviation, and correlation matrix estimated from the training data are shown in Table 8.7. As can be seen, the statistical properties of the training data are similar to the properties of the entire measurement time series, which we discussed in Section 8.2. Comparing non-robust and robust estimates, we observe again that the non-diagonal elements are larger in the case of the robust correlation matrix. Sample mean and M-estimator mean are slightly different whereas the standard deviation values do not differ much.

Table 8.8 shows the eigenvalues and the corresponding eigenvectors in the form of w-vectors for the sample correlation matrix and the M-estimator correlation matrix. The eigenvalues are also depicted in the scree plot of Figure 8.10. As can be seen, the eigenvalues are quite similar for both correlation matrices. There is one knee at the second PC and another at the forth PC. Hence, if we decided on the number of retained PCs using the scree test, we would keep the first two or the first four PCs.

A quantitative comparison of the non-robust and robust eigenvalues and PCs is problematic because they reflect differently standardized time series. Nevertheless, we observe that the differences are quite small. Although the absolute values of the loadings differ, the relative order of the standardized measurement variables with highest weights is the same for most PCs, independently of whether the non-robust or robust estimates are used. The

Tab. 8.7: Statistics of the training data

Mean and standard deviation

Metric	Mean		Standard deviation	
	non-robust	robust	non-robust	robust
Bytes	$4.2080 \cdot 10^8$	$3.5150 \cdot 10^8$	$2.5364 \cdot 10^8$	$2.4712 \cdot 10^8$
Packets	$8.6378 \cdot 10^5$	$7.3624 \cdot 10^5$	$4.6356 \cdot 10^5$	$4.5257 \cdot 10^5$
Flow records	$6.2194 \cdot 10^4$	$5.6394 \cdot 10^4$	$2.4348 \cdot 10^4$	$2.4830 \cdot 10^4$
Source IP addresses	$5.2408 \cdot 10^3$	$4.9404 \cdot 10^3$	$1.0728 \cdot 10^3$	$9.9599 \cdot 10^2$
Destination IP addresses	$5.3981 \cdot 10^3$	$5.0162 \cdot 10^3$	$1.1806 \cdot 10^3$	$1.0422 \cdot 10^3$
Source ports	$1.5747 \cdot 10^4$	$1.4448 \cdot 10^4$	$4.7730 \cdot 10^3$	$4.8481 \cdot 10^3$
Destination ports	$1.6480 \cdot 10^4$	$1.5219 \cdot 10^4$	$4.7489 \cdot 10^3$	$4.7918 \cdot 10^3$
Average flow duration	5.7305	5.7356	1.5424	1.5094

Sample correlation matrix

Bytes	Packets	Records	Src addr	Dst addr	Src port	Dst port	Avg dur
1.0000	0.9909	0.9160	0.8359	0.7394	0.9202	0.9213	−0.5427
0.9909	1.0000	0.9247	0.8497	0.7523	0.9342	0.9356	−0.5238
0.9160	0.9247	1.0000	0.7697	0.6847	0.9579	0.9541	−0.6387
0.8359	0.8497	0.7697	1.0000	0.9148	0.8267	0.8176	−0.3962
0.7394	0.7523	0.6847	0.9148	1.0000	0.7272	0.7227	−0.3645
0.9202	0.9342	0.9579	0.8267	0.7272	1.0000	0.9795	−0.5438
0.9213	0.9356	0.9541	0.8176	0.7227	0.9795	1.0000	−0.5158
−0.5427	−0.5238	−0.6387	−0.3962	−0.3645	−0.5438	−0.5158	1.0000

Robust M-estimator correlation matrix

Bytes	Packets	Records	Src addr	Dst addr	Src port	Dst port	Avg dur
1.0000	0.9947	0.9504	0.8698	0.8547	0.9470	0.9478	−0.5980
0.9947	1.0000	0.9543	0.8792	0.8640	0.9543	0.9551	−0.5798
0.9504	0.9543	1.0000	0.8209	0.8131	0.9709	0.9685	−0.6688
0.8698	0.8792	0.8209	1.0000	0.9712	0.8627	0.8564	−0.4428
0.8547	0.8640	0.8131	0.9712	1.0000	0.8501	0.8439	−0.4475
0.9470	0.9543	0.9709	0.8627	0.8501	1.0000	0.9853	−0.5915
0.9478	0.9551	0.9685	0.8564	0.8439	0.9853	1.0000	−0.5696
−0.5980	−0.5798	−0.6688	−0.4428	−0.4475	−0.5915	−0.5696	1.0000

Tab. 8.8: Eigenvalues and w-vectors

PCA applied to sample correlation matrix

i	1	2	3	4
Eigenvalue	6.5046	0.7805	0.4441	0.1399
Bytes	0.1481	−0.0027	−0.2725	1.6248
Packets	0.1493	−0.0375	−0.2951	1.3157
Records	0.1471	0.2047	−0.3312	−0.7574
Src addr	0.1382	−0.3996	0.5157	0.1075
Dst addr	0.1268	−0.4652	0.9037	−0.4095
Src port	0.1487	0.0274	−0.3767	−1.0224
Dst port	0.1480	−0.0003	−0.4454	−0.9830
Avg dur	−0.0928	−0.9278	−0.7465	−0.0961

i	5	6	7	8
Eigenvalue	0.0699	0.0338	0.0193	0.0076
Bytes	−0.4644	0.4112	−0.1963	7.5305
Packets	−0.3407	0.2200	−0.1645	−8.5828
Records	−0.8597	−4.3577	0.2807	0.4267
Src addr	2.8532	−1.2618	0.6193	0.2867
Dst addr	−2.1621	0.5820	−0.3012	−0.0036
Src port	0.5546	1.6943	−5.2199	0.3165
Dst port	0.1043	2.2547	4.8810	0.3036
Avg dur	−0.3029	−0.6710	−0.2029	0.3070

PCA applied to robust M-estimator correlation matrix

i	1	2	3	4
Eigenvalue	6.8392	0.7040	0.2960	0.0908
Bytes	−0.1428	−0.0142	0.3378	2.0049
Packets	−0.1433	−0.0514	0.3557	1.7038
Records	−0.1421	0.1538	0.5102	−0.6683
Src addr	−0.1340	−0.3911	−0.8884	−0.0747
Dst addr	−0.1327	−0.3832	−1.0044	−0.3569
Src port	−0.1428	−0.0118	0.4742	−1.3720
Dst port	−0.1423	−0.0387	0.5702	−1.2722
Avg dur	0.0938	−1.0451	0.7272	0.0121

i	5	6	7	8
Eigenvalue	0.0294	0.0219	0.0140	0.0044
Bytes	0.0302	−0.8826	−0.1276	9.9855
Packets	−0.0297	−0.3933	−0.3211	−11.130
Records	−2.1866	5.1515	0.3467	0.8050
Src addr	3.6920	2.4389	0.5016	0.2805
Dst addr	−3.6913	−1.7176	−0.1689	0.0370
Src port	1.0702	−1.7289	−6.0991	0.3935
Dst port	0.8174	−2.4151	5.7801	−0.0397
Avg dur	−0.3913	0.6159	−0.2023	0.4095

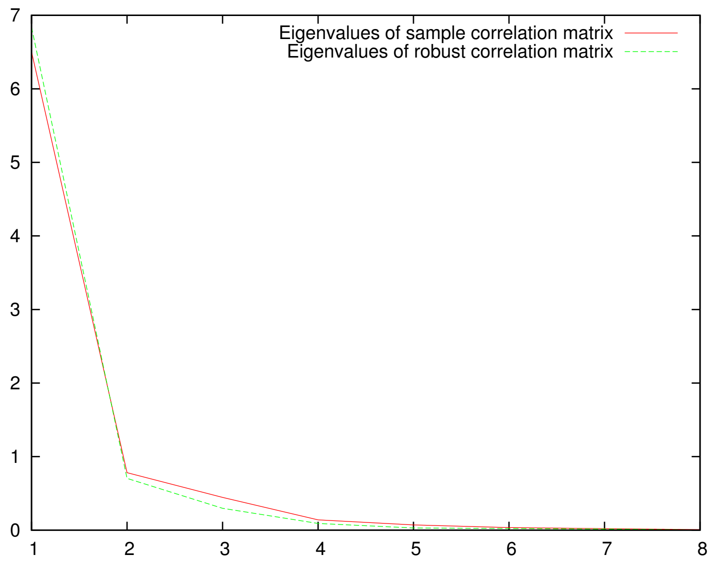


Fig. 8.10: Scree plot of batch-mode PCA

largest differences occur for PC5 and PC6.

Figures 8.11 and 8.12 display the y-score time series. The vertical lines indicate the end of the first two weeks which are used to estimate mean and covariance matrix. As can be seen, there is no big difference between the y-scores of non-robust and robust estimation, apart from a change of sign in some PCs.

PC1 covers the daily variation which is common to all metrics and contributes by far the largest part of variability in the data. As shown in Table 8.8, the loadings of non-robust and robust estimation are almost equal, yet with reversed signs. The sign of the average flow duration differs from the other measurement variables since the duration, in contrast to the other metrics, oscillates between low values at daytime and high values at nighttime.

PC2 gives the highest loadings to the average flow duration, followed by the numbers of distinct source and destination IP addresses. Anomalous values mainly occur around November 11, where we also observe an extremely large number of destination IP addresses in Figure 8.1. In addition to this very prominent anomaly, PC3 and PC5 show a change around October 6 when the mailbomb occurs. The time series of PC4 does not expose any large spikes. PC6, which is dominated by the numbers of records and source addresses, has one large peak on October 31 around 11pm in the night, where we discovered a large number of HTTP and DNS flow records in Section 8.3.3.

PC7 is essentially the difference between the numbers of source and destination ports, which are usually positively correlated as discussed in

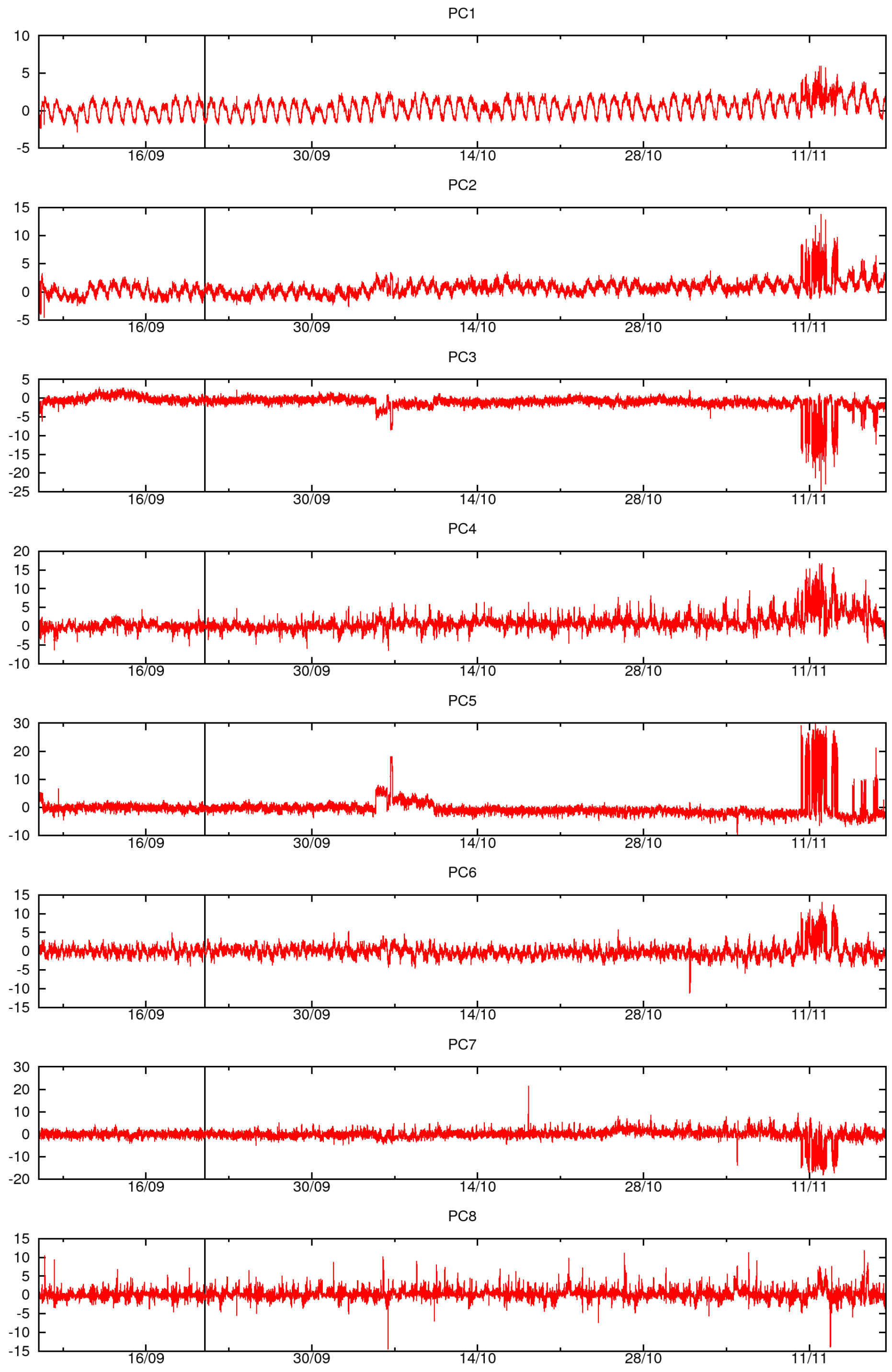


Fig. 8.11: y-scores of batch-mode PCA based on sample correlation matrix

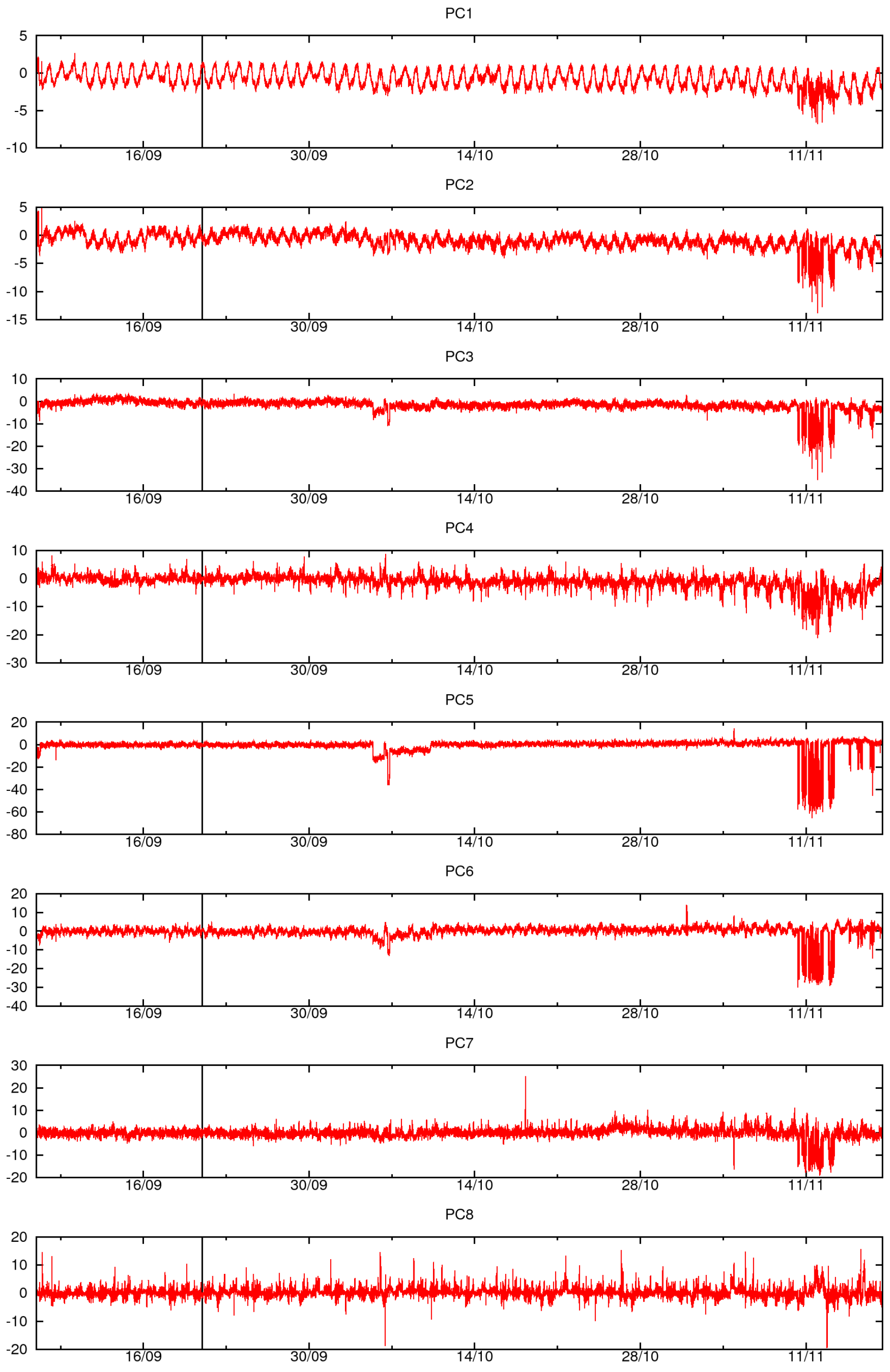


Fig. 8.12: y-scores of batch-mode PCA based on M-estimator correlation matrix

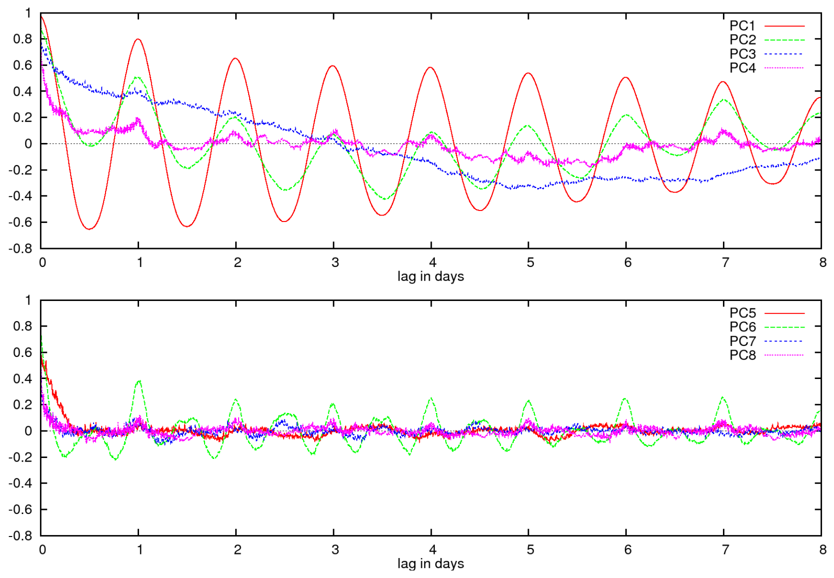


Fig. 8.13: Sample autocorrelation of y-scores in the training period (sample correlation matrix)

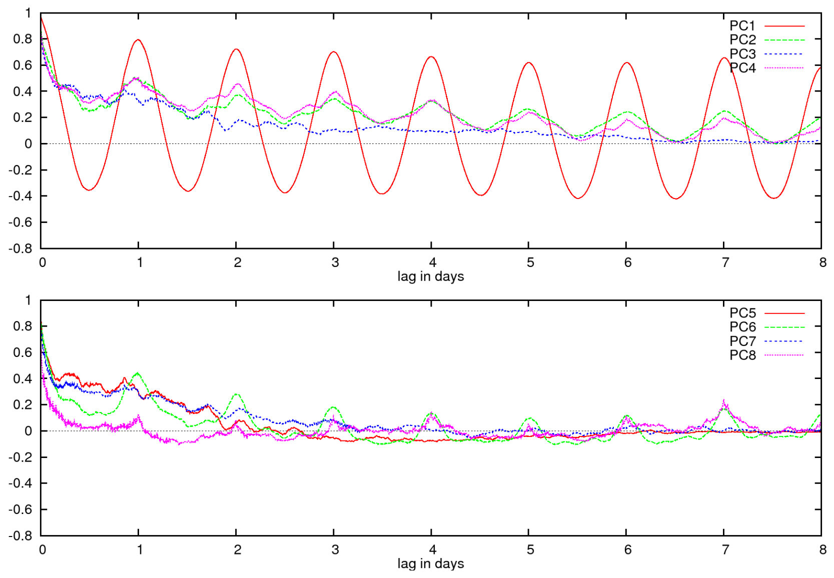


Fig. 8.14: Sample autocorrelation of y-scores in the detection period (sample correlation matrix)

Section 8.2. PC8 is the difference of the standardized byte and packet counts, again two metrics with positive correlation under normal conditions. The corresponding y-score time series look quite stationary and random with a couple of peaks where the usual correlation relationships seem to be violated.

The sample autocorrelation plots of the y-scores look very similar for the non-robust and robust case. Therefore, we only show the autocorrelation functions of the y-scores determined with sample mean and sample correlation matrix in Figures 8.13 and 8.14. Figure 8.13 depicts the sample autocorrelation for the first two weeks, which is the training period, confirming that seasonal variation is mainly present in the first PC. A certain dependency on the time of day is also present in PC2 whereas PC6 oscillates with double frequency and thus resembles a second harmonic. The remaining PCs are principally affected by serial correlation.

The sample autocorrelation plots in Figure 8.14 relate to the detection period (i.e., all but the first two weeks). The differences to Figure 8.13 indicate that the statistical properties of the y-scores do not remain the same. Stronger serial correlation causes a shift towards positive values in all PCs while the first PC is still dominated by daily variation. The long-lasting anomaly in November can be made responsible for this change.

The overall outcome of batch-mode PCA is not surprising. The daily variation which is common to all metrics is covered by the first PC, containing more than 80 percent of the entire variability. The PCs belonging to the smallest eigenvalues contribute random variability and a few isolated anomalies. In the y-score time series of the robust correlation matrix, some of the anomalies are more exposed than in the y-score time series of the sample correlation matrix. However, the differences between non-robust and robust estimation are small, probably because there are not any large anomalies in the two weeks of training data.

Change Detection Using T^2 and T^2_H Control Charts

Section 5.2.3 explained Hotelling's T^2-test as a method to detect outliers in multivariate data. The first and second plot in Figure 8.15 depict the time series of the T^2 statistic of non-robust and robust PCA. The third plot shows the T^2_H statistic which is calculated for the last four PCs only (i.e., PC5 to PC8). As mentioned in Section 5.2.4, the Hawkins' statistic is calculated like the T^2 considering the residual PC subspace only. T^2_H serves for testing if an observation can be adequately characterized by the retained PCs (i.e., PC1 to PC4). The y-axis is scaled logarithmically because the difference between the anomalous values around November 11 and the normal value range is extremely large.

As can be seen, the time series are very similar in shape. In particular, we find the same intervals with anomalously high values in all three plots. T^2 of robust PCA shows slightly higher values in anomaly intervals than the

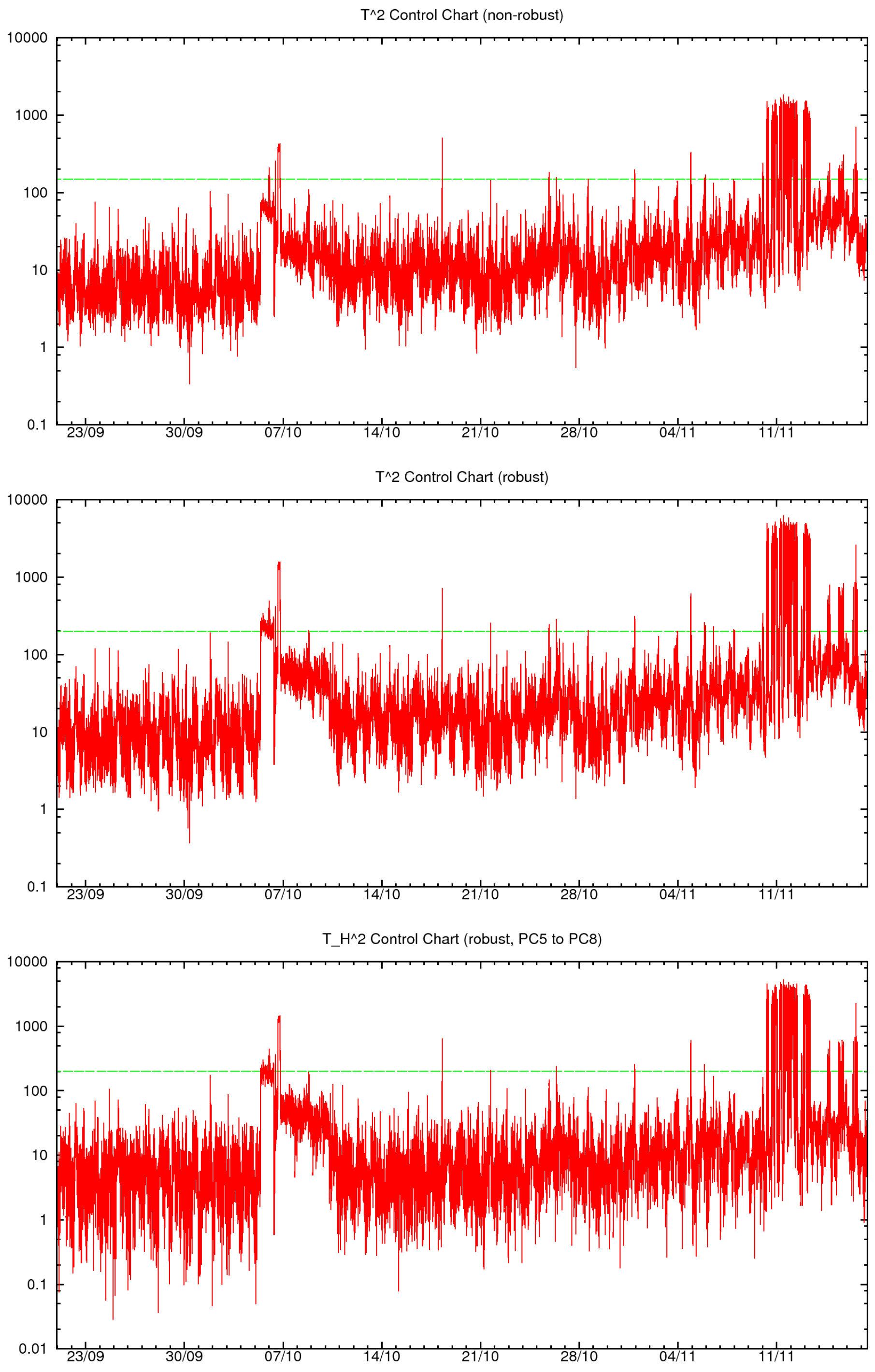


Fig. 8.15: T^2 and T_H^2 control charts

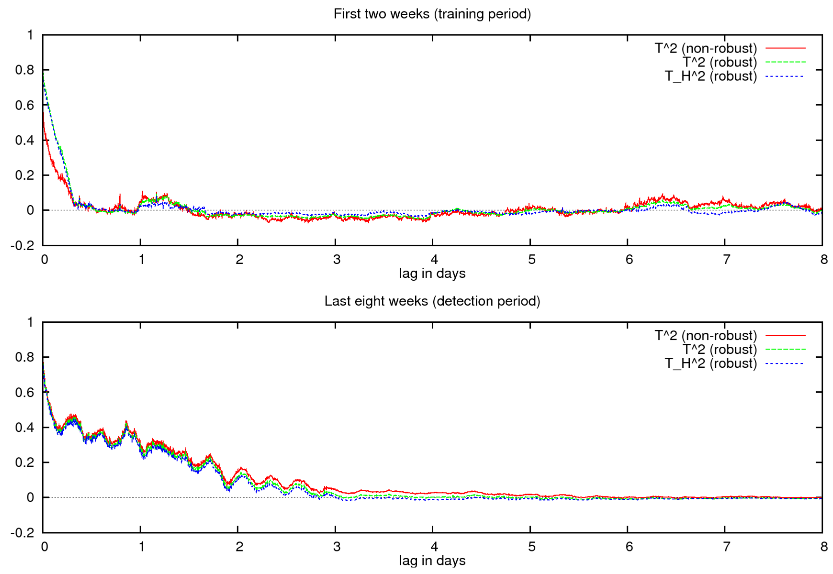


Fig. 8.16: Sample autocorrelation of T^2 and T_H^2 statistics

non-robust T^2 while the ranges of normal variation are more or less identical. This suggests that the detectability of anomalies might be improved with robust PCA. The similarity between T^2 and T_H^2 shows that anomalies in the T^2 time series are mainly caused by large values in the PCs of the residual subspace.

Figure 8.16 displays the sample autocorrelation functions of the three time series separately for the training period (first two weeks) and the detection period (last eight weeks). During the training period and for small lags, the sample autocorrelation of the robust T^2 and T_H^2 statistic is slightly larger than the corresponding values of the non-robust T^2 statistic. Apart from this, the curves are very similar in shape and magnitude. The most interesting observation is the absence of any periodic pattern which could be related to daily variation. This means that the calculation of the T^2 and T_H^2 statistics allows eliminating seasonality. It is even not necessary to omit the first PC which models the common seasonal variation because its influence on the T^2 statistic is very small compared to the effect of non-systematic changes which appear in multiple PCs.

To construct a control chart, an appropriate upper control limit is required. A lower control limit does not exist because of the quadratic form of T^2. The control limits introduced in Section 5.2.4 assume that the distribution of the original variables is multivariate normal, which is not fulfilled in our case. Hence, we have to set the control limit empirically.

A control limit of 150 is shown as a green horizontal line in the non-robust T^2 control chart in Figure 8.15. For the robust T^2 and T_H^2 control charts, we show the control limit at 200 which results in a comparable number

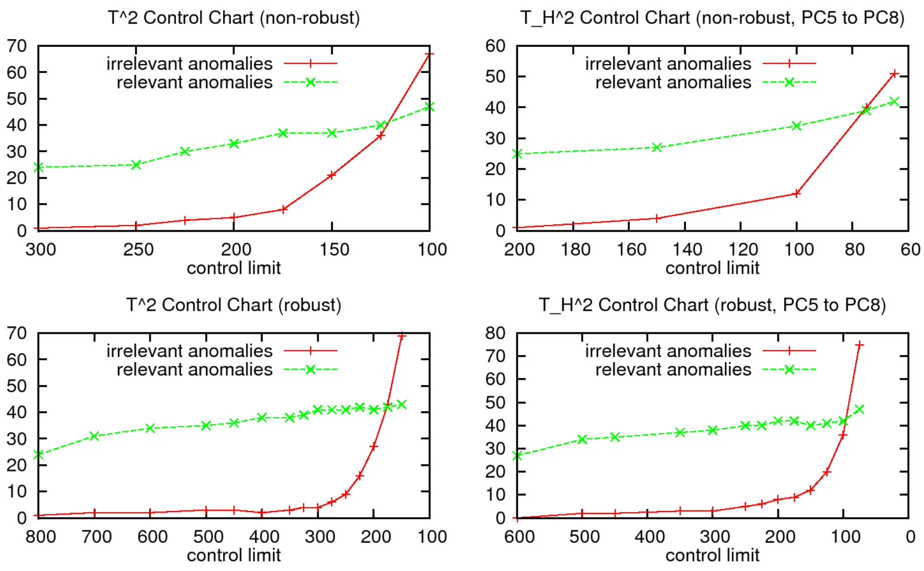


Fig. 8.17: Relevant and irrelevant anomalies of T^2 and T_H^2 control charts

of alarms. Between 37 (non-robust T^2) and 44 (robust T_H^2) of the detected anomalies are classified as relevant. At the same time, the robust T_H^2 control chart triggers alarms for 11 irrelevant anomalies. In the case of the non-robust and robust T^2 control charts, the numbers of irrelevant anomalies are 21 and 29, respectively.

Table 8.9 lists the times and causes of the relevant anomalies. Crosses in the column 'non-rob.' mark the anomalies detected with the non-robust T^2 control chart. The robust T^2 and T_H^2 control charts allow detecting the same set of relevant anomalies, which are marked in the column 'robust'. Most of the relevant anomalies were also discovered in the individual metrics with help of the Shewhart control chart (see Table 8.6). There are a few new alarms, mainly caused by ICMP network scans during the last days of the measurement period. Variation in DNS traffic is responsible for many irrelevant alarms. However, there are three DNS anomalies which are classified as relevant. On November 4, a DNS server stops responding twice; on November 12, a DNS server is flooded by many packets from two hosts.

The plots in Figure 8.17 show the numbers of relevant and irrelevant anomalies of non-robust and robust T^2 and T_H^2 control charts in dependence of the control limit. The number of detected anomalies increases for decreasing control limits. At a certain control limit, the number of irrelevant anomalies exceeds the number of relevant anomalies. At this point, we obtain about 40 relevant anomalies in all control charts.

Regarding Figure 8.17, there is no big difference between T^2 and T_H^2 control charts apart from a shift towards smaller control limits which can

Tab. 8.9: Relevant anomalies detected by T^2 and T_H^2 control charts

Time	Cause	non-rob.	robust
05/10 11:35 - 06/10 10:05	SMB network scan		X
05/10 12:00 - 05/10 12:00	MS-SQL network scan		X
06/10 01:45 - 06/10 04:00	SMTP (mailbomb)	X	X
06/10 08:05 - 06/10 08:05	VNC network scan		X
06/10 09:30 - 06/10 09:30	DCE network scan		X
06/10 10:05 - 06/10 10:20	SSH scan, password guessing		X
06/10 17:00 - 07/10 12:25	SMB network scan	X	X
18/10 08:20 - 18/10 08:45	Port scan	X	X
04/11 22:20 - 04/11 22:55	DNS server problem	X	X
04/11 23:10 - 04/11 23:10	DNS server problem		X
10/11 08:05 - 10/11 08:30	HTTPS network scan	X	X
10/11 08:50 - 10/11 09:10	HTTPS network scan (cont.)	X	X
10/11 09:35 - 10/11 11:35	HTTPS network scan (cont.)	X	X
10/11 16:25 - 10/11 18:00	HTTPS network scan (cont.)	X	X
10/11 18:10 - 11/11 00:50	HTTPS network scan (cont.)	X	X
11/11 01:30 - 11/11 01:45	HTTPS network scan (cont.)	X	X
11/11 05:35 - 11/11 10:20	HTTPS network scan (cont.)	X	X
11/11 12:20 - 11/11 12:25	HTTPS network scan (cont.)	X	X
11/11 12:35 - 11/11 14:25	HTTPS network scan (cont.)	X	X
11/11 14:40 - 11/11 19:40	HTTPS network scan (cont.)	X	X
11/11 20:20 - 11/11 21:20	HTTPS network scan (cont.)	X	X
11/11 22:05 - 12/11 01:45	HTTPS network scan (cont.)	X	X
12/11 02:00 - 12/11 06:50	HTTPS network scan (cont.)	X	X
12/11 09:00 - 12/11 10:00	HTTPS network scan (cont.)	X	X
12/11 10:45 - 12/11 12:10	HTTPS network scan (cont.)	X	X
12/11 19:20 - 12/11 19:45	DNS flood	X	X
12/11 22:40 - 13/11 06:35	HTTPS network scan (cont.)	X	X
13/11 07:30 - 13/11 09:05	HTTPS network scan (cont.)	X	X
13/11 09:30 - 13/11 09:55	HTTPS network scan (cont.)	X	X
13/11 10:05 - 13/11 10:05	HTTPS network scan (cont.)	X	X
14/11 16:15 - 14/11 17:10	ICMP network scan	X	X
14/11 18:10 - 14/11 19:15	ICMP network scan	X	X
15/11 10:00 - 15/11 10:50	ICMP network scan	X	X
15/11 11:40 - 15/11 12:30	ICMP network scan	X	X
15/11 14:35 - 15/11 14:50	ICMP network scan		X
15/11 15:55 - 15/11 17:40	ICMP network scan	X	
15/11 17:00 - 15/11 17:40	ICMP network scan	X	X
15/11 17:55 - 15/11 18:10	ICMP network scan		X
15/11 19:00 - 15/11 19:20	ICMP network scan	X	X
16/11 11:05 - 16/11 11:50	ICMP network scan	X	X
16/11 14:05 - 16/11 15:10	ICMP network scan	X	X
16/11 16:00 - 16/11 16:10	Two simultaneous ICMP scans	X	X
16/11 16:20 - 16/11 17:10	ICMP network scan	X	X
16/11 18:45 - 16/11 18:55	ICMP network scan	X	X

be explained by the fact that T_H^2 is always smaller than T^2 since it does not cover all PCs. Robust PCA results in larger T^2 and T_H^2 values, thus we need to use higher control limits than in the non-robust case in order to obtain similar results. Furthermore, we can obtain a larger proportion of relevant anomalies with the robust variants of the control charts. This is because the number of relevant anomalies stays close to 40 for a wide range of the control limits while the number of irrelevant anomalies remains smaller than five.

In summary, the utilization of robust M-estimators is advantageous since the same set of relevant anomalies can be detected with a smaller number of irrelevant anomalies. Moreover, we can obtain a large proportion of relevant anomalies over a wide range of control limits, which means that no sophisticated fine-tuning is required. The difference between Hotelling's T^2 statistic and Hawkins' T_H^2 statistic is negligible.

Change Detection in y-Score Time Series

Section 5.2.2 mentioned the possibility to apply control charts to individual PCs. As discussed in Section 8.4.1, PC1 covers most of the seasonal variation of the original variables while the remaining PCs are more or less free of systematic changes. Hence, we can apply the Shewhart control chart with constant control limits to the y-score time series of PC2 to PC8. For non-robust PCA, Figure 8.18 displays the corresponding numbers of relevant and irrelevant anomalies detected in dependence of the control limit. Except for PC8, we always find a setting where we detect more relevant than irrelevant anomalies.

Table 8.10 lists the causes of the relevant anomalies detected with control limits at ± 8. The bit vectors in the last two columns have the i-th bit set to 1 if the anomaly was found in the i-th PC, and 0 otherwise. The first bit, which is associated to PC1, is crossed out since PC1 is not considered in this evaluation. The results are almost identical to those obtained with the T^2 and T_H^2 control charts. Again, the different phases of the long-lasting HTTPS network scan in November account for a large number of anomalies.

As mentioned above, PC8 gives the highest loadings to the standardized byte and packet counts. Although this is the PC of the smallest eigenvalue, most of the alarms are associated to irrelevant traffic anomalies caused by large data transfers. This observation shows that there is no guarantee that a PC allows detecting relevant anomalies just because it belongs to the residual subspace.

We have repeated the experiment with the y-score time series of robust PCA achieving very similar results. Therefore, we do not discuss the results in more detail. In the next subsection, we apply exponential smoothing to the y-score time series and use the prediction errors as input to the Shewhart control chart.

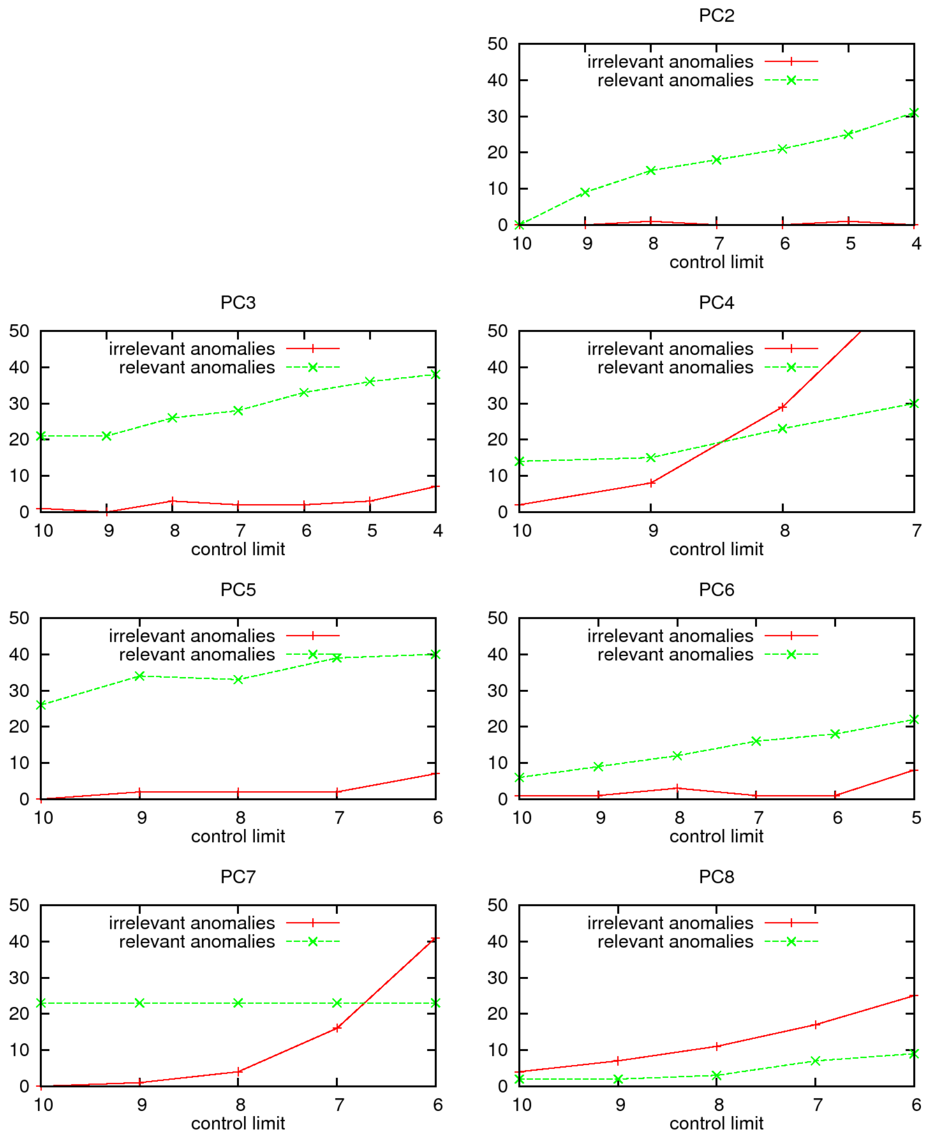


Fig. 8.18: Relevant and irrelevant anomalies detected in non-robust y-scores

Tab. 8.10: Relevant anomalies detected in non-robust y-scores (limit=8)

Time	Cause	PCs
06/10 01:45 - 06/10 04:00	SMTP (mailbomb)	x0000001
06/10 17:00 - 07/10 12:25	SMB network scan	x0101000
18/10 08:20 - 18/10 08:45	Port scan	x0000010
04/11 22:20 - 04/11 22:55	DNS server problem	x0001010
04/11 23:10 - 04/11 23:10	DNS server problem	x0000010
10/11 08:05 - 10/11 08:30	HTTPS network scan	x1101110
10/11 08:50 - 10/11 09:10	HTTPS network scan (cont.)	x0101010
10/11 09:35 - 10/11 11:35	HTTPS network scan (cont.)	x1111110
10/11 16:25 - 10/11 16:35	HTTPS network scan (cont.)	x0111010
10/11 17:20 - 10/11 18:00	HTTPS network scan (cont.)	x0111010
10/11 18:10 - 11/11 00:50	HTTPS network scan (cont.)	x1111110
11/11 01:30 - 11/11 01:45	HTTPS network scan (cont.)	x0111110
11/11 05:35 - 11/11 10:20	HTTPS network scan (cont.)	x1111110
11/11 12:20 - 11/11 12:25	HTTPS network scan (cont.)	x1111010
11/11 12:35 - 11/11 14:25	HTTPS network scan (cont.)	x1111010
11/11 14:40 - 11/11 19:40	HTTPS network scan (cont.)	x1111110
11/11 20:20 - 11/11 21:20	HTTPS network scan (cont.)	x1111110
11/11 22:05 - 12/11 01:45	HTTPS network scan (cont.)	x1111110
12/11 02:00 - 12/11 06:50	HTTPS network scan (cont.)	x1111110
12/11 09:00 - 12/11 10:00	HTTPS network scan (cont.)	x1111110
12/11 10:45 - 12/11 12:10	HTTPS network scan (cont.)	x1111010
12/11 19:20 - 12/11 19:45	DNS flood	x0000001
12/11 22:40 - 13/11 06:35	HTTPS network scan (cont.)	x1111110
13/11 07:30 - 13/11 09:05	HTTPS network scan (cont.)	x1111110
13/11 09:30 - 13/11 09:55	HTTPS network scan (cont.)	x1111010
13/11 10:05 - 13/11 10:05	HTTPS network scan (cont.)	x0101010
14/11 00:45 - 14/11 00:45	RAdmin network scan	x0010000
14/11 01:10 - 14/11 01:35	SSH network scan, password guessing	x0010000
14/11 01:45 - 14/11 01:45	TCP port 2120 network scan	x0010000
14/11 02:45 - 14/11 02:45	HTTP network scan	x0010000
14/11 16:15 - 14/11 17:10	ICMP network scan	x0010000
14/11 18:10 - 14/11 19:15	ICMP network scan	x0001000
15/11 10:00 - 15/11 10:50	ICMP network scan	x0001000
15/11 11:40 - 15/11 12:30	ICMP network scan	x0101000
15/11 15:55 - 15/11 17:40	ICMP network scan	x0101001
15/11 17:00 - 15/11 17:40	ICMP network scan	x0001000
15/11 19:00 - 15/11 19:20	ICMP network scan	x0011000
16/11 11:05 - 16/11 11:50	ICMP network scan	x0001000
16/11 14:05 - 16/11 15:10	ICMP network scan	x0101000
16/11 16:00 - 16/11 16:10	Two ICMP network scans	x0101000
16/11 16:20 - 16/11 17:10	ICMP network scan	x0101000
16/11 18:45 - 16/11 18:55	ICMP network scan	x0001000

Combining Batch-Mode PCA and Exponential Smoothing Forecasting

PCA ensures that the PCs are mutually uncorrelated, yet the corresponding y-score time series are not entirely free of systematic changes and serial correlation as we have seen in Figures 8.14. Systematic changes dominate the y-score time series of PC1, yet they also appear in various other PCs with smaller magnitude. Serial correlation is present in almost all PCs. Consequently, control charts and statistical tests which assume stationarity or even independence of observations risk to trigger more false alarms than expected.

In order to reduce temporal dependencies, we apply exponential smoothing with $\alpha = 0.5$ to the y-score times series as a second residual generation step. The resulting residual time series are depicted in Figure 8.19. The residual time series of all PCs, including PC1, now resemble the output of a pure random process. In the autocorrelation function (which is not shown here), we see that serial correlation as well as seasonal effects have been largely eliminated.

After applying a Shewhart control chart with constant control limits at ± 5, we obtain the long list of relevant alarms shown in Table 8.11. As before, the bit vectors in the last two columns indicate by 1s and 0s if the anomaly was found in the corresponding PC or not. The residuals of PC1 do not trigger any relevant or irrelevant alarms with this setup, which means that there are not any large changes. Since we use constant control limits, several alarms are triggered at the end of an anomaly, which also causes another abrupt change in y-score time series. Just like in Section 8.3.4, we account these alarms as relevant anomalies.

Figure 8.20 shows the numbers of relevant and irrelevant anomalies detected in every PC in dependence of the control limits. Again, most of the anomalies found in PC8 are irrelevant. Compared to Figure 8.18, we obtain larger numbers of relevant anomalies, yet these principally go back to the alarms triggered at the end of an anomaly. As before, irrelevant alarms are mainly caused by data transfers (HTTP, FTP, RSF-1) and DNS traffic.

We roughly evaluated how the results differ if the robust y-score time series are used as input. As in the preceding experiments, the differences are marginal. Therefore, we omit a presentation of the results.

Discussion

Batch-mode PCA relies on training data which is used to estimate the means and covariances of the original measurement variables. It transforms the original variables into uncorrelated PCs. As a result, common seasonal variation is modeled by the PC of the largest eigenvalue. The remaining PCs are far less affected by systematic changes, yet serial correlation remains

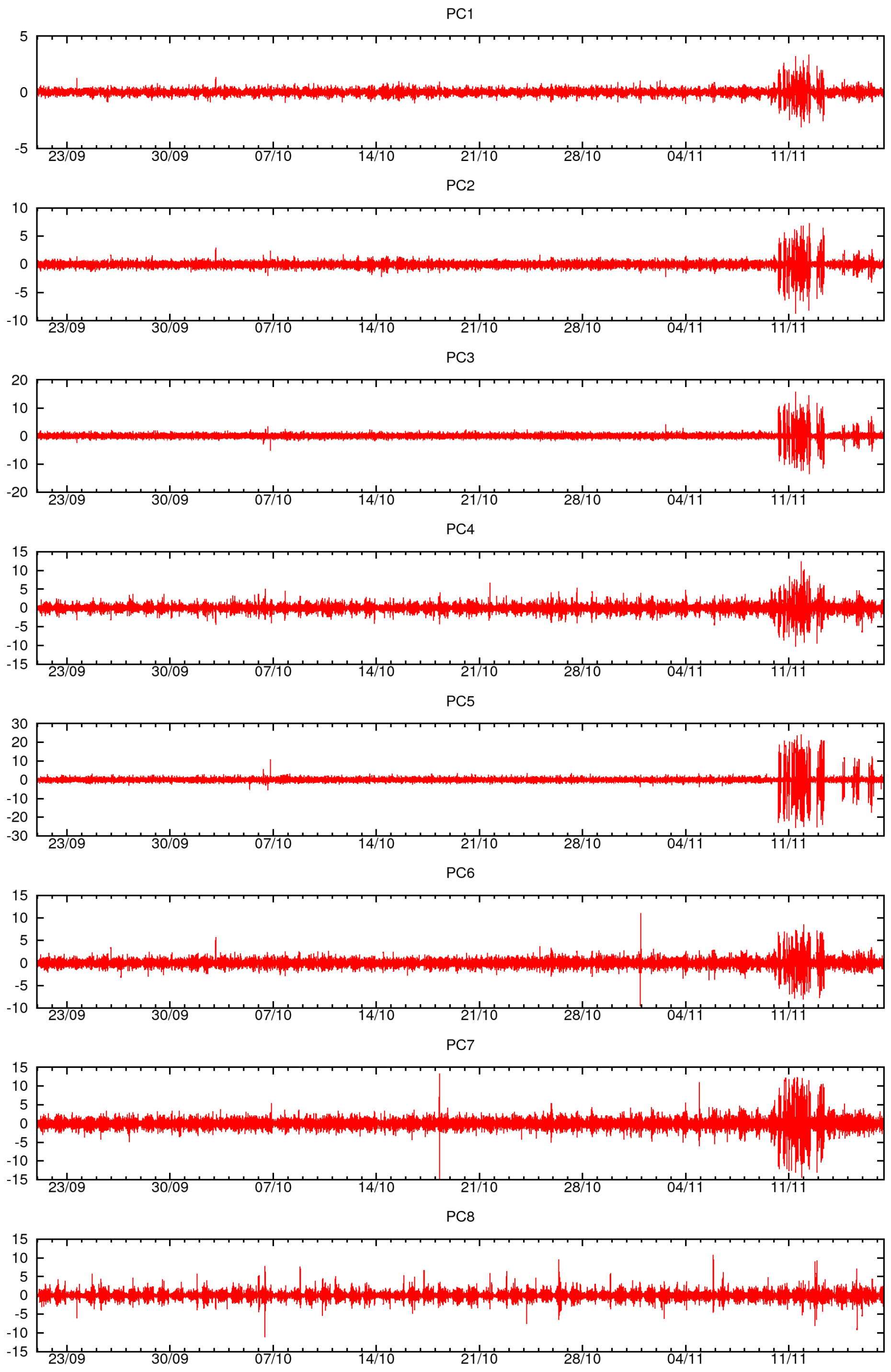


Fig. 8.19: ES residuals of y-scores of batch-mode PCA (non-robust)

Tab. 8.11: Relevant anomalies detected in ES residuals of y-scores

Time	Cause	PCs
23/09 17:50 - 23/09 18:15	SSH password guessing	00000001
03/10 04:20 - 03/10 05:40	SSH password guessing	00000100
06/10 01:45 - 06/10 04:00	SMTP (mailbomb)	00000001
06/10 10:10 - 06/10 10:10	end of SMB network scan	00001000
06/10 13:15 - 06/10 13:20	SMTP	00010000
06/10 17:00 - 07/10 12:25	SMB network scan	00001010
10/10 10:00 - 10/10 10:30	Increased syslog traffic	00000001
18/10 08:20 - 18/10 08:45	Port scan	00000010
02/11 13:25 - 02/11 14:20	ICMP ping flood	00000001
04/11 23:00 - 04/11 23:05	end of DNS server problem	00000010
10/11 08:05 - 10/11 08:30	HTTPS network scan	00101110
10/11 08:50 - 10/11 09:10	HTTPS network scan (cont.)	00100100
10/11 09:15 - 10/11 09:20	end of HTTPS network scan	00011010
...	...	...
12/11 10:45 - 12/11 12:10	HTTPS network scan (cont.)	00101110
12/11 12:15 - 12/11 12:20	end of HTTPS network scan	00011010
12/11 19:20 - 12/11 19:45	DNS flood	00000001
12/11 22:30 - 13/11 06:35	HTTPS network scan (cont.)	00111110
13/11 06:40 - 13/11 06:45	end of HTTPS network scan	00011010
13/11 07:30 - 13/11 09:05	HTTPS network scan (cont.)	00101100
13/11 09:10 - 13/11 09:20	end of HTTPS network scan	01011010
13/11 09:30 - 13/11 09:55	HTTPS network scan (cont.)	00100100
13/11 10:00 - 13/11 10:00	end of HTTPS network scan	01011010
13/11 10:10 - 13/11 10:15	end of HTTPS network scan	00001010
14/11 17:15 - 14/11 17:15	end of ICMP network scan	00001000
14/11 18:10 - 14/11 19:15	ICMP network scan	00001000
14/11 19:20 - 14/11 19:20	end of ICMP network scan	00001000
14/11 19:45 - 14/11 19:50	end of ICMP network scan	00001010
15/11 10:00 - 15/11 10:50	ICMP network scan	00001000
15/11 11:40 - 15/11 12:30	ICMP network scan	00001000
15/11 14:35 - 15/11 14:50	ICMP network scan	00001000
15/11 14:55 - 15/11 14:55	end of ICMP network scan	00001000
15/11 15:55 - 15/11 17:40	ICMP network scan	00001000
15/11 16:35 - 15/11 16:40	ICMP network scan	00001000
15/11 17:00 - 15/11 17:40	ICMP network scan	00001000
15/11 17:55 - 15/11 18:10	ICMP network scan	00001000
15/11 19:00 - 15/11 19:20	ICMP network scan	00001000
16/11 11:05 - 16/11 11:50	ICMP network scan	00101000
16/11 11:55 - 16/11 11:55	end of ICMP network scan	00001000
16/11 14:05 - 16/11 15:10	ICMP network scan	00001000
16/11 16:00 - 16/11 16:10	ICMP network scans	00101000
16/11 17:15 - 16/11 17:15	end of ICMP network scan	00001000
16/11 18:45 - 16/11 18:55	ICMP network scan	00001000

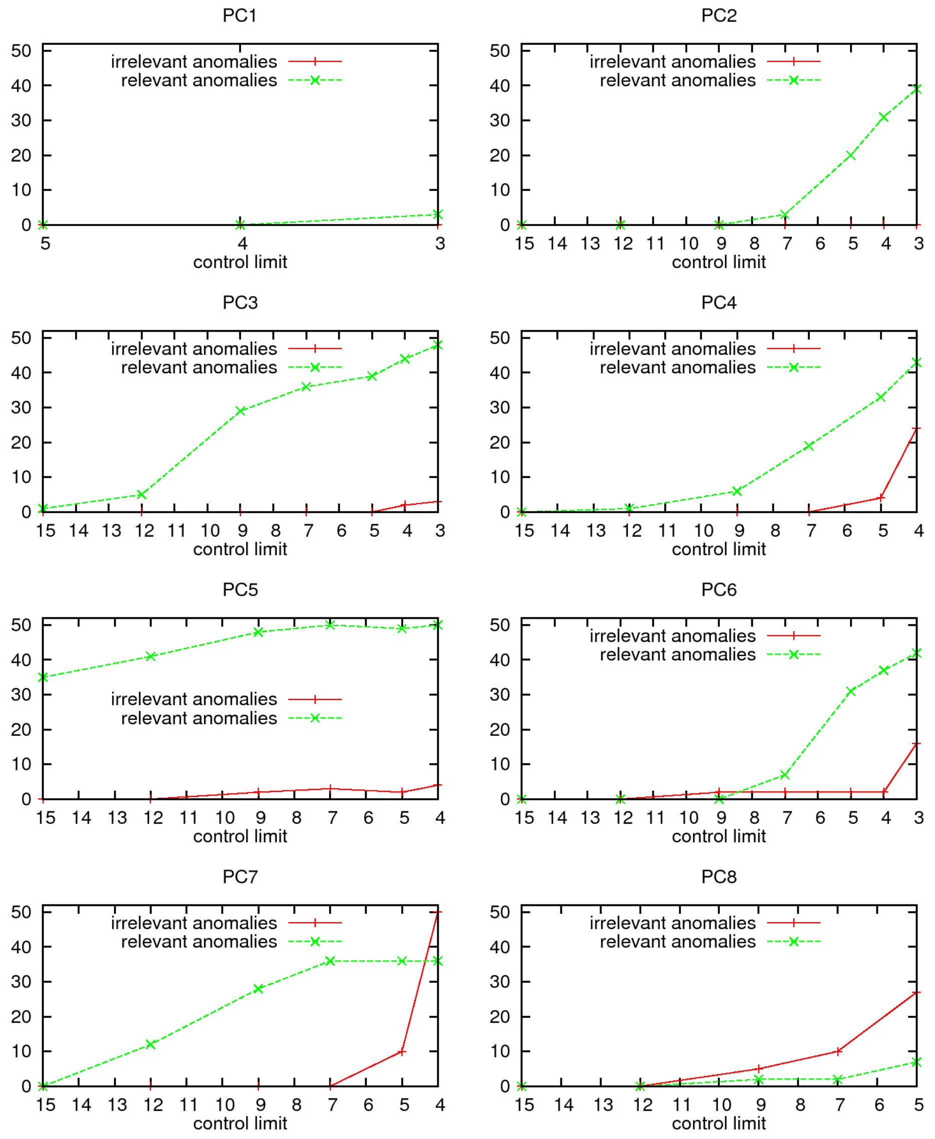


Fig. 8.20: Relevant and irrelevant anomalies detected in ES residuals of non-robust y-scores

present in the sample autocorrelation functions.

Changes can be detected by applying the Shewhart control chart with constant upper control limit to Hotelling's T^2 statistic, reflecting all PCs, or Hawkins' T_H^2 statistic, reflecting only the residual PCs of the smallest eigenvalues. With this approach, we detected many relevant anomalies over a wide range of control limits while the number of irrelevant anomalies was low. Batch-mode PCA based on robust estimates of the mean and covariances turned out be superior to the non-robust version since fewer irrelevant alarms were triggered.

As an alternative to T^2 and T_H^2 control charts, changes can be detected in the y-score times series of the individual PCs. For those PCs which are not affected by systematic changes, the Shewhart control chart can be directly applied to the y-score time series. Otherwise, it is possible to deploy exponential smoothing as an additional residual generation step, which reduces temporal dependencies. Applied to our measurement data, change detection in individual PCs did not reveal any new kinds of anomalies. While the detection results were similar, we lost the advantage of the T^2 and T_H^2 statistic of having only a single control chart for all metrics.

We only presented the detection results of control charts with constant control limits. The same kinds of control charts can be deployed with adaptive control limits based on moving estimators of the standard deviation. As an advantage, adaptive control limits facilitate the configuration of the thresholds which are defined as multiples of the standard deviation. Regarding the detected anomalies, the effect is similar as in the case of single-metric anomaly detection. Certain additional relevant anomalies are found while others are missing. The number of irrelevant anomalies is larger than in the case of constant control limits.

All in all, the Shewhart control chart of the T^2 or T_H^2 statistic is the most appropriate and simplest change detection solution for batch-mode PCA. However, it is recommended to use robust estimators in order to reduce the bias caused by outliers in the training data. The proportion of relevant anomalies is larger than in most of the single-metric control charts. On the other hand, the majority of the relevant anomalies was also detected by the single-metric approach, which means that taking into account the correlation between different metrics in the residual generation process does not lead to the detection of novel kinds of anomalies. Network scans are once again responsible for the majority of the relevant alarms. As discussed in Section 8.3.3, the number of distinct destination IP addresses is an excellent metric to detect these kinds of anomalies without causing a lot of irrelevant alarms.

A general problem of PCA-based traffic anomaly detection is that we get no or little information about which original measurement variables are the most affected by an anomaly. This is particularly true for the T^2 and T_H^2 statistics. Hence, searching the original flow data for anomaly causes is more difficult than in the single-metric case where the metric in which an anomaly is detected already gives an indication of possible causes.

8.4.2 Incremental PCA

Batch-mode PCA has the disadvantage that the modeled correlation structure sticks to the training data from which the covariance or correlation matrix is obtained and does not adapt to gradual changes of the traffic. As discussed in Section 5.5, we can overcome these limitations by using exponentially weighted moving estimators to estimate means and covariances of the time series. In this case, the estimated covariance matrix is updated with every new observations, thus enabling the recalculation of the eigenvalues and w-vectors. This approach is known as incremental PCA in the literature.

Implementation

We implemented incremental PCA based on the following moving EWMA and EWMC estimators for the mean $\overline{x}$ and the covariance matrix $\mathbf{S}$:

$$\overline{x}_t = \theta_1 \mathrm{x}_t + (1 - \theta_1)\overline{x}_{t-1}$$

$$\mathbf{S}_t = \theta_2(\mathrm{x}_t - \overline{x}_t)(\mathrm{x}_t - \overline{x}_t)' + (1 - \theta_2)\mathbf{S}_{t-1}$$

With identical smoothing constant $\theta = \theta_1 = \theta_2$, both estimators have the same exponential decay with time. Initial values are calculated from the first N time-series values $\mathrm{x}_1, \ldots, \mathrm{x}_N$ using the sample mean and sample covariance matrix:

$$\overline{x}_N = \frac{1}{N}\sum_{t=1}^{N} \mathrm{x}_t \quad ; \quad \mathbf{S}_N = \frac{1}{N-1}\sum_{t=1}^{N}(\mathrm{x}_t - \overline{x}_p)(\mathrm{x}_t - \overline{x}_p)'$$

$N \geq p+1$ values are required, p being the number of measurement variables, in order to obtain a sample covariance matrix of full rank. Since the initial values are quite inaccurate, a couple of additional observations and estimator updates are necessary until $\overline{x}_t$ and $\mathbf{S}_t$ approach reliable values. For our evaluation, we calculate the initial values from the first $N = 100$ time-series values and perform 188 recursive updates before we determine the principal components for the first time. Hence, the y-score time series start on the second day of the measurement period.

As before, we standardize the original measurement variables and calculate the PCs from the correlation matrix. For standardizing the i-th variable at time t, we use the EWMA value $\bar{x}_{t-1,i}$ as mean and the square roots of the diagonal elements of $\mathbf{S}_{t-1}$ as standard deviations $\hat{\sigma}_{t-1,i}$:

$$(x_{t,i} - \bar{x}_{t-1,i})/\hat{\sigma}_{t-1,i}$$

Tab. 8.12: Final eigenvalues and w-vectors of incremental PCA with $\theta = 0.001$

i	1	2	3	4	5	6	7	8
Eigenvalue	5.4335	1.1168	0.7634	0.4306	0.1434	0.0541	0.0382	0.0195
Bytes	0.1486	0.4223	0.1938	0.6990	0.2582	1.2188	0.4999	4.1844
Packets	0.1586	0.3970	0.1553	0.4133	0.3862	−0.5483	−0.2890	−5.3277
Records	0.1767	−0.0153	−0.2488	−0.1131	0.2165	−3.4824	−0.8185	2.1182
Src addr	0.1519	0.2010	0.3819	−0.8097	−1.7246	0.3618	−0.1870	0.1629
Dst addr	0.0923	−0.5371	0.8011	0.1922	0.4068	−0.5358	1.4992	−0.0728
Src port	0.1696	−0.2585	−0.0483	−0.3332	0.9783	1.7423	−3.3016	0.0526
Dst port	0.1643	−0.0357	−0.4485	−0.5371	0.7074	0.8715	3.4541	−0.4608
Avg dur	−0.1347	0.4024	0.4414	−0.7466	1.4523	−0.5699	0.0095	0.7599

The correlation matrix is obtained from the estimated covariance matrix.

With every recursive update of the covariance matrix, the PCs are recalculated, resulting in time-variant eigenvalues and w-vectors. The scaling as w-vectors defines the length of the eigenvectors, but not their orientation. Therefore, the recalculation of the PCs may lead to arbitrary sign changes in the PCs and the corresponding y-scores. To prevent this effect, we define that the loading of the first original variable (i.e, the number of bytes) is always non-negative.

Residuals Generation Using Incremental PCA

It is difficult to compare the time-variant PCs of incremental PCA with the invariant PCs of batch-mode PCA. As an example, Table 8.12 shows the final eigenvalues and w-vectors that result after the last time-series value is processed. The smoothing constant has been set to $\theta = 0.001$ in order to inhibit fast changes of the estimates. The first eigenvalue is smaller than in the case of batch-mode PCA (see Table 8.8), meaning that the variability is less concentrated in the first PC. Nevertheless, the first eigenvector resembles PC1 of batch-mode PCA. The last two PCs (PC7 and PC8) are also similar to the batch-mode result, whereas the remaining PCs do not reveal any similarities.

We applied incremental PCA with different settings for the smoothing constant θ in order to assess the influence of this parameter on the results. Figure 8.21 shows the y-score time-series plots for $\theta = 0.001$. The time series of the first PC bears a strong resemblance to that of batch-mode PCA (see Figure 8.11) while all the other time series are very different. In the y-scores of batch-mode PCA, we find some anomalies that appear as sustained shifts of the mean. EWMA and EWMC ensure that the estimated mean and covariances adapt to such changes in the original measurements time series. Therefore, these anomalies appear as short spikes in the y-scores of incremental PCA.

Figure 8.22 displays the sample autocorrelation functions of the y-score time series obtained with $\theta = 0.001$. As for batch-mode PCA, PC1 reveals strong seasonality while the periodic pattern is much less exposed in the other PCs. The effect of serial correlation is weak.

Incremental PCA can be regarded as a multivariate extension of univariate exponential smoothing. While EWMA provides prediction values of the mean, EWMC forecasts the covariance matrix in the next time interval. In Section 8.3, $\theta = 0.5$ turned out to be a good smoothing constant for exponential smoothing. If we try the same setting for incremental PCA, we obtain y-scores time series which are almost completely free of any serial correlation. The corresponding time-series plots and the sample autocorrelation functions are depicted in Figures 8.23 and 8.24. The y-score time series resemble the output of pure random processes. We verified if the PCs can still be considered as uncorrelated variables. In fact, the correlation between different y-scores is similar for $\theta = 0.001$ and $\theta = 0.5$. None of the values exceeds 0.1. Thus, incremental PCA with $\theta = 0.5$ efficiently reduces correlation among different metrics and temporal dependencies simultaneously.

As can be seen in Figure 8.23, the value ranges of the y-scores are much larger than for smaller values of θ. The standard deviation of the different y-score time series ranges from 2.1 for PC1 up to 40.7 for PC8 whereas it is between 1.0 and 1.1 for $\theta = 0.001$. Thus, the y-scores cannot be assumed to have unit variance any more if θ is large. The eigenvalues range from 5.6 for PC1 to only 0.000045 for PC8, which means that the eigenvalues of the last PCs are very small. This is not a disadvantage as long as we analyze the y-score time series individually. However, the formulas presented in Sections 5.2.3 and 5.2.4 should not be used any more to calculate T^2 and T^2_H statistics from y-scores for large θ.

Change Detection using T^2 and T^2_H Control Charts

Figure 8.25 displays the T^2 statistic calculated from the y-scores of incremental PCA with $\theta = 0.001$. Compared to the T^2 time series of batch-mode

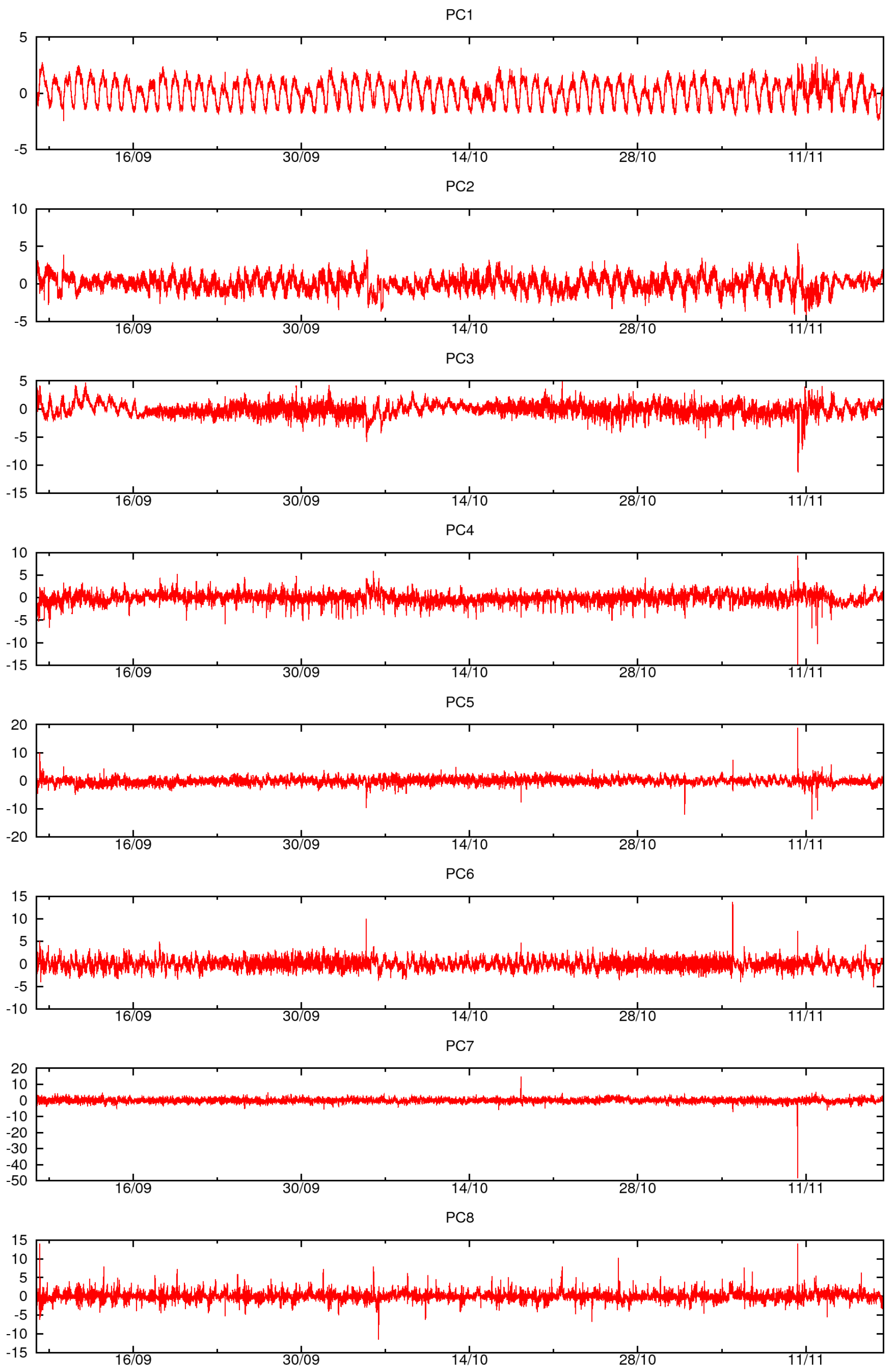


Fig. 8.21: y-scores of incremental PCA with $\theta = 0.001$

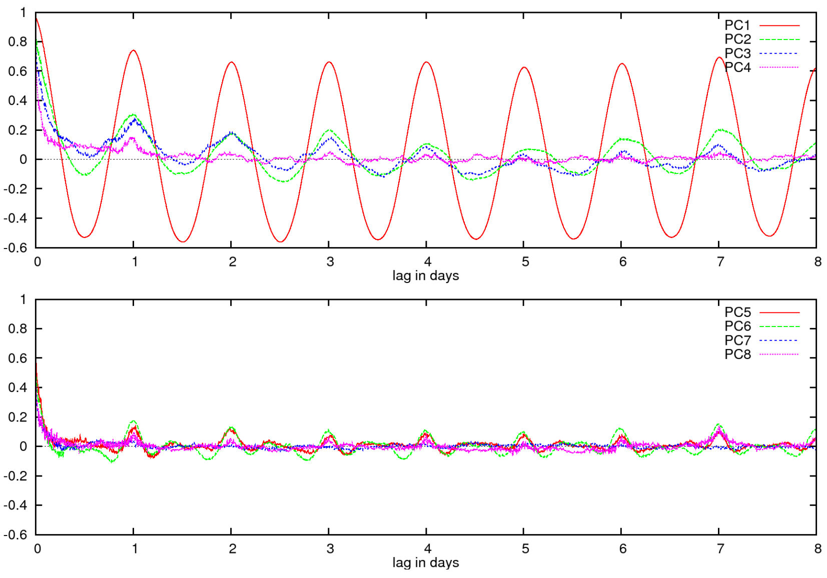


Fig. 8.22: Sample autocorrelation function of y-scores ($\theta = 0.001$)

PCA, the smoothing effect is obvious. All shifts which last for a longer period of time in Figure 8.15 are much smaller now and diminish over time.

Applying the Shewhart control chart with constant upper control limit at 70, we obtain 26 relevant and 11 irrelevant alarms, which corresponds to a proportion of relevant alarms of 70%. The irrelevant alarms are principally related to HTTP and FTP data transfers as well as RSF-1 and DNS traffic. Lowering the threshold to 50, we get 35 relevant alarms, 23 irrelevant alarms, and a relevant proportion of only 60%. With batch-mode PCA, we were able to detect more than twice as many relevant alarms at the same level of irrelevant alarms (see Section 8.4.1).

The Shewhart control chart has also been applied to the T^2_H statistic calculated from PC5 to PC8. In this case, we have obtained worse results with more irrelevant than relevant alarms. Hence, the only advantage of incremental PCA compared to batch-mode PCA seems to be that we do not need any training data to estimate the correlation matrix.

Change Detection in y-Score Time Series

We applied the Shewhart control chart with constant control limits to the y-score time series of incremental PCA with $\theta = 0.5$. Since the y-score value range varies a lot among the different PCs, the control limits have to be set to very different levels as well. In general, the y-score value range increases with decreasing eigenvalue (i.e., according to the order of the PCs). For example, a control limit of ± 200 already causes 42 alarms in PC8 whereas not a single alarm is triggered in PC1 to PC5.

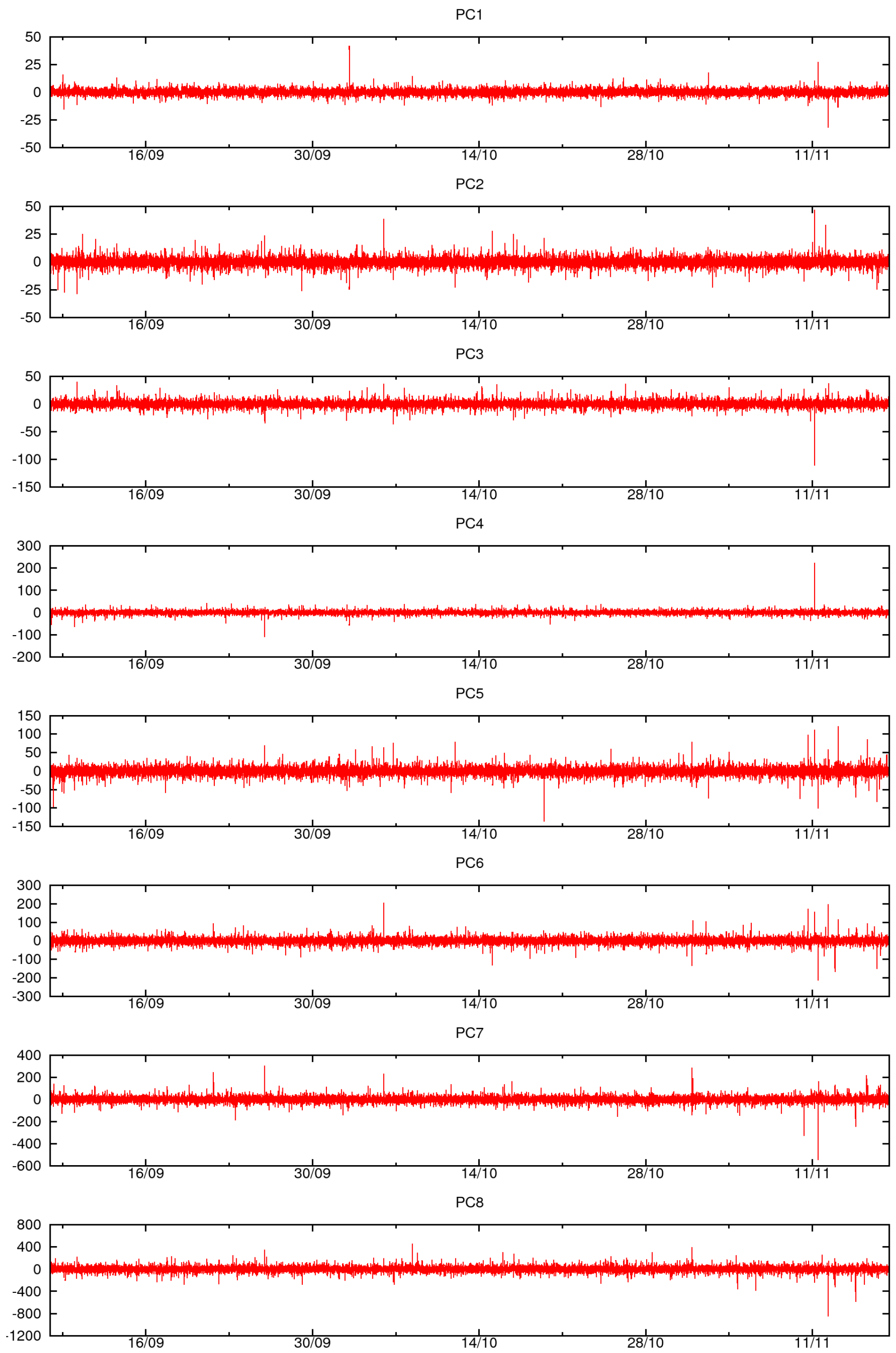


Fig. 8.23: y-scores of incremental PCA with $\theta = 0.5$

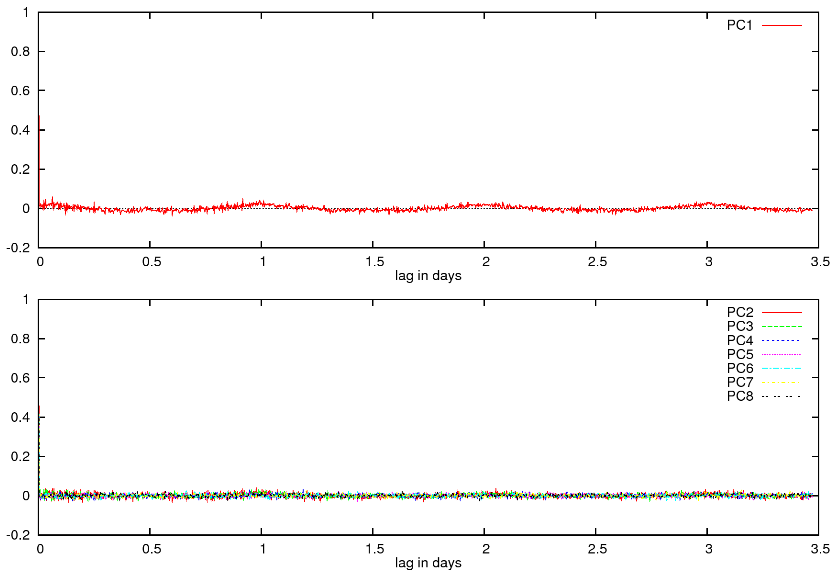


Fig. 8.24: Sample autocorrelation function of y-scores ($\theta = 0.5$)

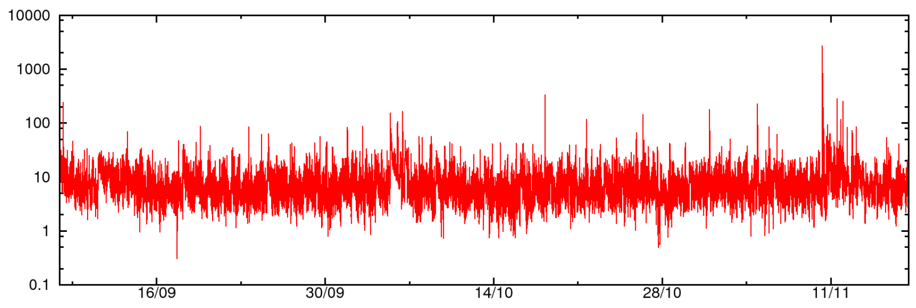


Fig. 8.25: T^2 of incremental PCA with $\theta = 0.001$

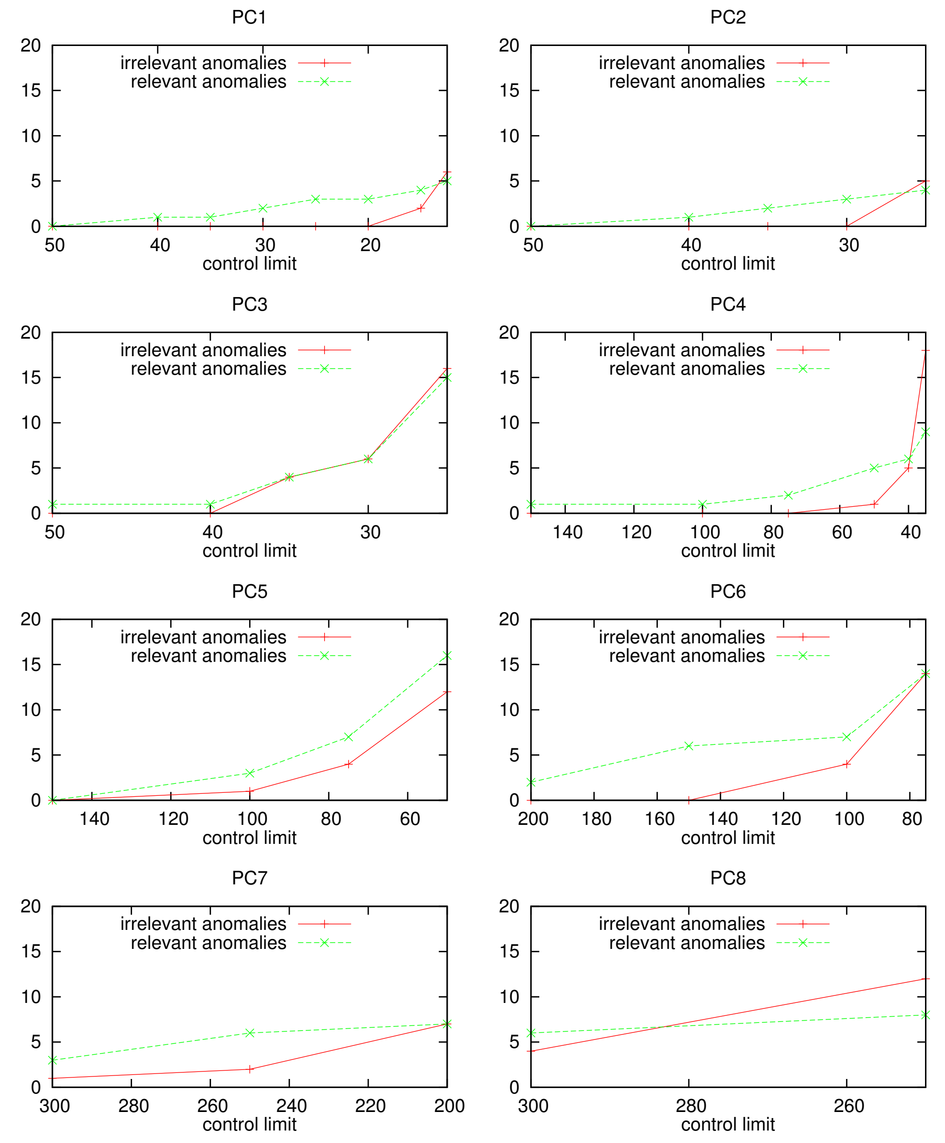


Fig. 8.26: Relevant and irrelevant anomalies detected in y-scores of incremental PCA ($\theta = 0.5$)

Figure 8.26 shows the numbers of anomalies detected in different PCs and for different control limits. In the depicted intervals, the gap between the numbers of irrelevant and relevant anomalies is small compared to what we have seen in the case of batch-mode PCA (see Figures 8.18 and 8.20). At the control limit where the irrelevant alarm count starts exceeding the relevant alarms, we get a rather small amount of relevant alarms.

Regarding the causes of the relevant alarms, we find the same events as before: a lot of network scans, a few port scans, passwort guessing attempts, network problems, and the mailbomb. Many of the irrelevant alarms are due to periodic increases of UDP traffic on ports 1026 and 1027 (Calendar Access Protocol) resulting in large numbers of distinct source and destination IP addresses and source ports. A service of the Windows operation system uses these ports to exchange messages between hosts. Although this service can be misused to distribute spam, we assume that the observed traffic is legitimate and harmless as it appears very frequently. Apart from that, DNS traffic and data transfers cause a lot of irrelevant alarms.

Discussion

Incremental PCA does not require any training data as it is based on moving estimators for the mean and covariance matrix. This is an advantage over batch-mode PCA, where the quality of the training data has a significant impact on the PCs. Furthermore, batch-mode PCA produces a time-invariant model which does not adapt to changes in the normal traffic. Incremental PCA can be tuned to generate y-score time series that are mutually uncorrelated and do not exhibit any systematic changes or serial correlation. In order to obtain similar residuals with batch-mode PCA, a second residual generation step is necessary to eliminate the remaining temporal dependencies between consecutive time-series values.

As a downside, incremental PCA updates the PCs with every new observation, which drastically increases the computational complexity. In addition, appropriate values for the smoothing parameters need to be found and configured. After all, the change detection results are not convincing. With batch-mode PCA, a quite large number of relevant anomalies could be detected while only few irrelevant anomalies disturbed the result. With incremental PCA, the numbers of relevant and irrelevant anomalies are much closer to each other, yielding smaller proportions of relevant anomalies.

8.5 Discussion of Results

In this chapter, we evaluated the applicability of the residual generation and change detection techniques presented in Chapters 3, 4 and 5 for detecting traffic anomalies in measurement time series. Therefore, we considered the time series of eight different metrics characterizing the overall IP traffic observed at a router in an ISP backbone network. Apart from evaluating the generated residuals from a statistical point of view (e.g., with help of the sample autocorrelation function), we identified the causes of the detected anomalies in order to classify them as relevant or irrelevant, depending on the potential importance for the network administrator. Although this classification is subjective to a certain extend, it allows us to compare the detection results of the examined anomaly detection methods.

Regarding the different residual generation and change detection techniques, the results of our experiments show that more complex approaches do not necessarily yield better results than simple methods. For single-metric anomaly detection, simple exponential smoothing turned out to be as appropriate as Holt-Winters to cope with seasonal variation and serial correlation in the original measurement variables. Regarding the change detection methods, we did not find any benefits of using the CUSUM and EWMA control charts instead of the simple Shewhart control chart of individuals. Finally, PCA did not lead to the detection of new types of relevant anomalies which could not be detected by single-metric anomaly detection as well. Hence, the question whether the additional complexity of PCA is worthwhile cannot be answered easily.

It must be denoted that the forecasting techniques are not able to produce residuals which are completely stationary and uncorrelated under normal traffic conditions. The same is true for the T^2 and T_H^2 statistics calculated with PCA. Hence, the necessary conditions for making any assertions regarding the false positive rate and other properties of the change detection methods are not fulfilled. Nevertheless, our evaluation shows that control charts are useful tools for detecting different kinds of relevant traffic anomalies.

In the following, we summarize the pros and cons of the evaluated single-metric and multi-metric anomaly detection approaches.

Single-metric anomaly detection based on robust time-series forecasting and control charts is easy to implement and parameterize. If the forecasting values are determined with exponential smoothing, there is a single smoothing constant α to configure. In our evaluation, $\alpha = 0.5$ turned out to be a good setting for all measurement variables. No training data is required since the forecasting is based on the EWMA of preceding observation. Updating the EWMA estimate with every new observation is a non-complex operation. Holt-Winters forecasting comes along with two additional smoothing constants for the trend and seasonal components, which, however, did not improve the forecasting result.

The Shewhart control chart of individuals compares every new residual value with the given control limits. Hence, apart from the control limits, no additional information is needed to make the decision. We tested this control chart with constant and adaptive control limits. In order to define constant control limits, we need to know or estimate the value range and

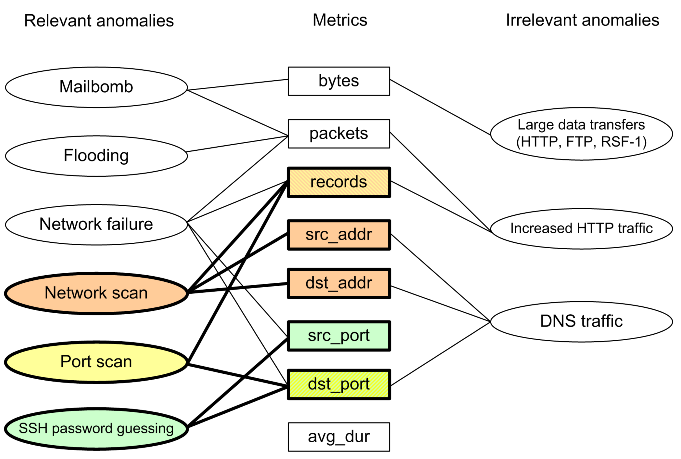


Fig. 8.27: Common relevant and irrelevant anomalies and affected metrics

variation under normal traffic conditions. Adaptive control limits automatically adapt to the variation in the residual time series, which facilitates the configuration since the limits are defined as multiples of the standard deviation estimated by EWMS or an alternative moving estimator. Updating the adaptive control limits requires another non-complex operation for every new observation.

The more sophisticated CUSUM and EWMA control charts have additional parameters, namely the reference value and the EWMA smoothing constant, respectively. In theory, these two control charts are more sensitive to small sustained changes than the Shewhart control chart. In practice, we could not profit from this property since such small but long-lasting changes did not appear in the time series of prediction errors.

The relevance of the detected anomalies significantly depends on the considered traffic metric. Large proportions of relevant anomalies can be achieved by analyzing cardinality metrics. On the other hand, most of the anomalies detected in volume metrics are irrelevant.

Figure 8.27 shows the relationship between frequent anomaly causes and the affected metrics. Network scans and SSH password guessing are the most frequent causes of relevant anomalies, followed by port scans, flooding, mailbombs, and network failures. Most irrelevant anomalies are caused by large data transfers and irregular fluctuations in HTTP and DNS traffic. Network scans principally influence the number of distinct destination IP addresses and, to a smaller extend, the numbers of distinct source IP addresses and flow records. SSH password guessing causes many distinct source and destination ports. Port scans lead to an increased number of records and destination ports. Traffic fluctuations caused by DNS influence the record, address, and port counts, too, yet usually with lower intensity. Flooding and mailbombs are difficult to distinguish from data transfers because all these events primarily affect the numbers of bytes and packets.

For the purpose of multi-metric anomaly detection, we applied batch-mode and incremental PCA to the measurement time series. Batch-mode PCA estimates the mean and covariances of the original measurement variables using training data. Since the sample mean and covariances are very susceptible to outliers in the training data, we evaluated the utilization of robust M-estimators as an alternative. From a statistical point of view, the non-robust and robust residuals were very similar. With respect to the detection results, robust control charts of Hotelling's T^2 and Hawkins' T_H^2 statistic produced a larger proportion of relevant anomalies over a wide range of control limits. On the other hand, the difference between T^2 and T_H^2 was negligible.

Incremental PCA relies on moving estimators of the mean and covariances and therefore does not require any training data. As a downside, the PCs are continuously updated, which entails an expensive recalculation of eigenvalues and eigenvectors with every new observation. Although incremental PCA eliminated correlation among different metrics as well as temporal dependencies, the detection results were rather disappointing since many of the anomalies were classified irrelevant.

Compared to single-metric anomaly detection, PCA allows monitoring all metrics with a single T^2 or T_H^2 control chart. However, the identification of anomaly causes is more difficult since we do not know directly which metrics are the most affected. All in all, the T^2 and T_H^2 control charts of batch-mode PCA enable the detection of many relevant anomalies with only a few irrelevant alarms and thus represent good alternatives to single-metric anomaly detection. We recommend the utilization of robust M-estimators, which has improved the detection results in our evaluation.

With single-metric methods, some of the relevant anomalies, such as network scans, could be detected very well while others, such as mailbombs, were only found at the cost of many irrelevant alarms. In these cases, the detection with multi-metric methods can be more efficient. Therefore, it makes sense to combine single-metric anomaly detection in cardinality metrics with multi-metric anomaly detection to reduce the risk of missing an important incident while keeping the number of irrelevant alarms small.

In the next chapter, we apply single-metric and multi-metric anomaly detection methods to specific parts of traffic instead of the overall IP traffic.

9. ANOMALY DETECTION IN SPECIFIC PARTS OF TRAFFIC

9.1 Introduction

In Chapter 8, we applied various residual generation and change detection methods to the time series of the overall IP traffic measured in an ISP backbone network. The combination of exponential smoothing and Shewhart control chart turned out to be an appropriate single-metric anomaly detection method. Furthermore, we achieved very good results with multi-metric anomaly detection based on batch-mode PCA and the T^2 and T_H^2 control charts. As a general problem, however, many of the alarms were classified as irrelevant. Irrelevant anomalies are mainly caused by unpredictable changes in the traffic of certain applications, such as HTTP and DNS.

This chapter deals with the analysis of specific parts of traffic which we expect to be less affected by spontaneous traffic changes under normal conditions. The corresponding flow records need to be separated from the flows of the remaining traffic. If the traffic to be analyzed is characterized by specific flow key values, the extraction of the flows can be easily achieved with a filter as mentioned in Section 7.3. As an example, if we want to analyze traffic of an application which uses a well-known port number, we can generate time series for those flow records having this port number in the source or destination port field.

Of course, port-based filtering fails if the same port is used by another application than expected. Furthermore, it is not possible to identify flow records of protocols or applications which cannot be associated with specific ports. In such a case, however, it is still possible to remove the traffic of other applications which are known to cause irrelevant alarms, such as HTTP or DNS traffic. Hence, there are several ways how filters can help to reduce the number of irrelevant alarms and to detect anomalies which are too small to be found in the overall IP traffic.

In the following two sections, we analyze the time series of ICMP (Internet Control Message Protocol) and SMB (Server Message Block) traffic and present the results of single-metric and multi-metric anomaly detection. Section 9.4 concludes this chapter with a discussion of the results.

9.2 Anomaly Detection in ICMP Traffic

As part of the Internet protocol suite, the Internet Control Message Protocol (ICMP) is mainly used for exchanging error messages, for example, if a certain host cannot be reached due to link or routing problems. ICMP is also used for network testing and debugging purposes (e.g., using *ping* and *traceroute* commands) and self configuration in local IP networks. As ICMP is not directly involved in the transport of user and application data, we expect a low and rather invariant level of ICMP traffic under normal network conditions. Furthermore, we do not expect any significant changes in the ICMP traffic due to normal network usage.

9.2.1 ICMP Traffic Characteristics

Figures 9.1 and 9.2 show the time series and the sample autocorrelation functions for the number of bytes, packets, flow records, distinct source and destination IP addresses as well as for the average flow duration of the recorded ICMP traffic. As expected, the time series values remain at a low level, apart from a few intervals with much larger values.

The byte count does not show any seasonal variation at all. The dependency on the time of day is also very low for the number of packets. In contrast, the sample autocorrelation functions of the numbers of records, source and destination IP addresses reveal significant seasonal variation with a period of one day. There is no increase at the lag of one week in any of the autocorrelation functions, which means that ICMP traffic is very similar on weekdays and weekends. The average flow duration shows by far the largest serial correlation among all the metrics.

There are a few isolated peaks in the time-series plots of the numbers of bytes, packets, and source IP addresses. Furthermore, we observe multiple very high time-series values in the record count and the number of distinct destination IP addresses between November 12 and the end of the measurement. At the same time, the average flow duration shows anomalously low values. The reasons for these anomalies are discussed in the next sections.

Table 9.1 presents the sample correlation matrix of the original variables, calculated over the entire measurement time. As can be seen, the byte count is correlated with the packet count. Also, the number of records is highly correlated with the number of destination IP addresses. On the other hand, the correlation between the numbers of distinct source and destination IP addresses is relatively low, which indicates that ICMP traffic is often unidirectional. In Section 9.2.3, we will see that the correlation matrix looks different if it is calculated from the first two weeks of data only. Apparently, the large anomalies in November have a strong influence on the correlation.

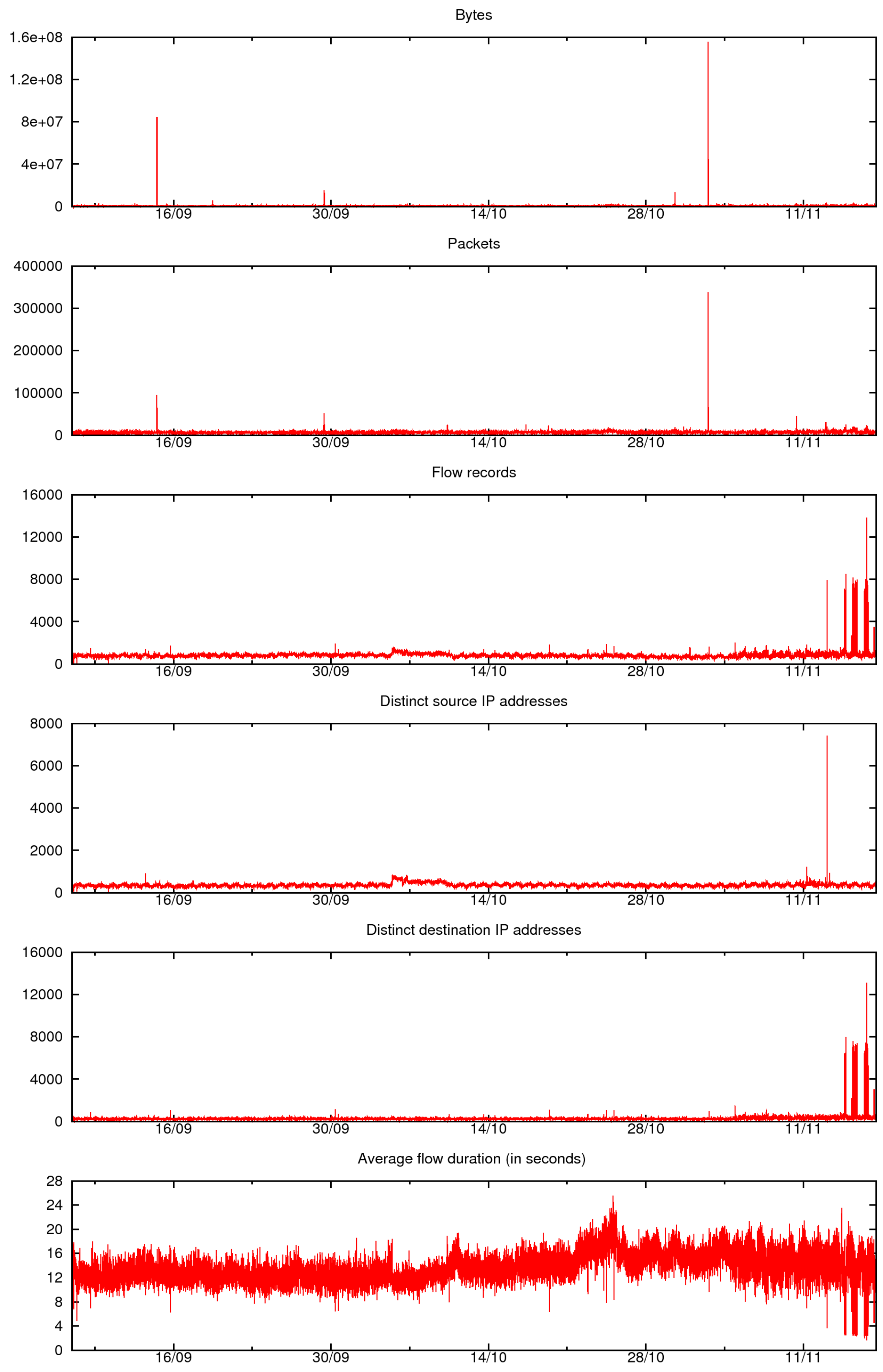


Fig. 9.1: Time series of ICMP traffic

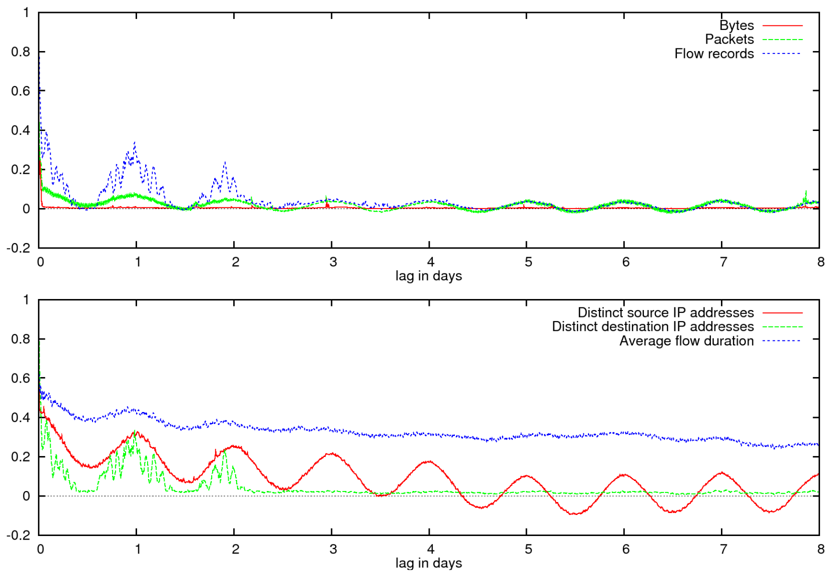


Fig. 9.2: Sample autocorrelation of ICMP traffic

Tab. 9.1: Sample correlation matrix of ICMP traffic

Bytes	Packets	Records	Src addr	Dst addr	Avg dur
1.0000	0.4930	0.0417	0.0265	0.0416	0.0196
0.4930	1.0000	0.2617	0.1347	0.2435	−0.0026
0.0417	0.2617	1.0000	0.3021	0.9629	−0.4693
0.0265	0.1347	0.3021	1.0000	0.0775	−0.2827
0.0416	0.2435	0.9629	0.0775	1.0000	−0.3672
0.0196	−0.0026	−0.4693	−0.2827	−0.3672	1.0000

9.2.2 Single-Metric Analysis of ICMP Traffic

We adopt the most promising single-metric residual generation and change detection approach of Section 8.3, which is calculating the prediction errors of exponential smoothing with smoothing constant $\alpha = 0.5$ and applying the Shewhart control chart of individuals with adaptive control limits to the residual time series. As before, the control limits are based on the EWMS moving estimator of the standard deviation ($\rho = 0.01$).

Control limits at $\pm 6\hat{\sigma}$, as used in Section 8.3.3, generate an extremely large number of alarms for byte and packet counts. Therefore, we start with an evaluation of the detection results achieved with control limits at $\pm 8\hat{\sigma}$ for all metrics. With this setup, we get 75 alarms in 48 different time intervals, most of them found in the number of bytes. No alarms are triggered by the average flow duration.

Table 9.2 lists all detected anomalies and their causes. The bit vector in the last column indicates by 1s in which metrics the anomaly is detected, using the following ordering: bytes, packets, records, source addresses, destination addresses. The anomalies can be grouped into three categories according to the metrics in which they are detected: anomalies detected in the byte or packet count, anomalies detected in the number of distinct source IP addresses, and anomalies detected in the number of records and distinct destination IP addresses. These three patterns can be explained as follows:

- Byte and packet counts are mainly affected by large ICMP flows exchanged between two hosts. Reasons for such flows are ping traffic and ICMP messages returned due to other packets which could not be delivered to the destination (message type 'time exceeded' and 'fragmentation required') or which should be sent on another path ('redirect').

- An increase of the number of source IP addresses is linked to ICMP port unreachable messages which are sent from a large number of IP addresses to a single host. These ICMP messages are related to network scanning activities originating from this host. Usually, ICMP port unreachable messages are returned in the case of a closed UDP port. However, we observe these messages during a network scan on TCP port 443 (HTTPS).

- The numbers of flows and destination IP addresses increase mostly during ICMP network scans using ICMP echo request messages. On November 4, the same pattern occurs when a DNS server suddenly closes its UDP port 53 for a short period of time; incoming DNS queries are then answered by ICMP port unreachable messages.

Apart from very few exceptions, all alarms fit into these three categories.

Tab. 9.2: Anomalies detected in ICMP traffic

Time	Cause	Metrics
10/09 07:15 - 10/09 07:15	Network failure	00100
11/09 09:00 - 11/09 09:05	Low rate ping (echo replies only)	10000
12/09 15:15 - 12/09 15:15	Low rate ping (echo requests only)	10000
13/09 14:45 - 13/09 14:45	Port unreachable from various addresses	00110
14/09 13:50 - 14/09 13:50	Ping flood (more than 73.000 packets)	11000
14/09 14:25 - 14/09 14:25	Ping flood (more than 83.000 packets)	10000
14/09 14:40 - 14/09 14:40	Ping flood (more than 57.000 packets)	10000
15/09 19:40 - 15/09 19:40	ICMP scan, followed by TCP 3389	00101
19/09 14:00 - 19/09 14:05	Moderate rate ping (echo replies only)	11000
21/09 15:30 - 21/09 15:40	Fragmentation required, low rate ping	11000
24/09 08:25 - 24/09 08:35	Fragmentation required	10000
29/09 10:40 - 29/09 11:20	High rate ping	11000
29/09 11:30 - 29/09 11:30	High rate ping (echo replies only)	10000
30/09 11:05 - 30/09 11:05	ICMP scan, followed by TCP 1433	00101
05/10 16:35 - 05/10 16:40	Fragmentation required	10000
06/10 18:00 - 06/10 18:10	Fragmentation required	10000
10/10 10:00 - 10/10 10:30	Time exceeded (TCP 514)	11000
13/10 07:50 - 13/10 07:55	Port unreachable (high port numbers)	01000
16/10 16:45 - 16/10 16:45	Fragmentation required	10000
17/10 09:55 - 17/10 10:05	Moderate rate ping	11000
19/10 09:30 - 19/10 09:35	Moderate rate ping (only echo requests)	11000
19/10 12:20 - 19/10 12:20	ICMP scan, followed by TCP 3389/1433	00101
21/10 23:00 - 21/10 23:15	Fragmentation required	10000
24/10 13:20 - 24/10 13:20	ICMP scan, followed by TCP 3389	00101
25/10 05:35 - 25/10 05:40	ICMP scan, TCP 80/8080/3128	00101
25/10 15:35 - 25/10 15:45	Low rate ping	10000
28/10 11:10 - 28/10 11:15	Fragmentation required	10000
30/10 09:20 - 30/10 09:20	Fragmentation required	10000
30/10 15:00 - 30/10 15:00	High rate ping (echo requests only)	11000
31/10 09:10 - 31/10 09:20	High rate ping (echo replies only)	01000
31/10 22:35 - 31/10 23:55	Low rate pings, measurement artefact?	00100
02/11 13:25 - 02/11 14:20	Ping flood	11000
02/11 15:50 - 02/11 15:50	ICMP scan, followed by TCP 5110	00101
04/11 22:55 - 04/11 22:55	Port unreachable (DNS server problem)	00101
07/11 20:00 - 07/11 20:05	Low rate ping (echo requests only)	10000
09/11 06:40 - 09/11 06:50	Low rate ping (echo replies only)	01000
10/11 09:10 - 10/11 09:10	Time exceeded, host unr. (HTTPS scan)	11000
10/11 10:25 - 10/11 10:25	Redirect from host/network (HTTPS)	11000
11/11 07:50 - 11/11 07:50	Port unreachable (HTTPS scan)	00010
12/11 23:55 - 13/11 00:40	Time exceeded (HTTPS scan)	11000
13/11 03:20 - 13/11 03:25	Port unreachable (HTTPS scan)	00110
13/11 11:00 - 13/11 11:00	Low rate ping (echo requests only)	10000
14/11 16:15 - 14/11 17:10	ICMP network scan	01101
15/11 10:00 - 15/11 10:15	ICMP network scan	00101
16/11 11:05 - 16/11 11:50	ICMP network scan	00101

Regarding the relevance of the alarms, those caused by ping traffic at low and moderate rate are the least interesting for the network administrator. Therefore, we do not consider these incidents as relevant. Ping traffic at high rate can be a sign of a flooding attack although it may also be the result of measurement experiments. We classify such high rate pings as relevant.

Most of the ICMP destination unreachable messages reporting 'fragmentation required' are directed to the same host which apparently runs a web server. Hence, such messages may be the result of path MTU discovery mechanisms determining the maximum packet size which can be exchanged between two hosts without fragmentation. Curiously, the ICMP destination unreachable messages accumulate in a few intervals. Nevertheless, we classify these alarms as irrelevant.

In contrast, all alarms related to scanning activities are considered as relevant. Quite often, we observe the following behavior. One hosts scans a few hundred addresses which all belong to the same subnet of the ISP. 100 to 170 active hosts answer with ICMP echo reply messages. The scanning host then tries to establish TCP connections to the active hosts using ports of Microsoft Windows services (e.g., Windows-based terminal on port 3389 and MS SQL server on port 1433) or web proxy ports (80, 8080, and 3128). It is difficult to assess if the observed traffic is harmful or not because such scans may serve testing and debugging purposes. However, it has also been reported that ICMP scans are more and more frequently deployed by malware and worms in advance of an infection attempt [Dac08]. There are several other ICMP network scans which last for a long time and probe a very large number of destinations. In these cases, we do not observe any TCP connection attempts originating from the scanning host.

Finally, we observe various ICMP traffic patterns related to TCP scanning activities. For example, during the long lasting HTTPS scan in November (e.g., see Table 8.6), we detect several anomalies due to an increased number of ICMP time exceeded, redirect, and port unreachable messages. These are all caused by TCP SYN packets which do not reach their destinations.

Figure 9.3 shows the distribution of the number of total and relevant alarms among the different metrics. The proportion of relevant alarms ranges from 53% for the number of bytes to 100% for the numbers of distinct source and destination IP addresses. Hence, as in Chapter 8, cardinality metrics result in a larger proportion of relevant alarms than volume metrics.

If we lower the control limits to $\pm 6\hat{\sigma}$, the average flow duration still does not trigger any alarm. Many more alarms are found in the number of bytes and packets. Regarding the number of records, source and destination IP addresses, the amount of alarms increases to 22, 7, and 16, respectively. The reasons for the new alarms are essentially the same as discussed before.

All in all, anomalies found in the ICMP traffic give the network ad-

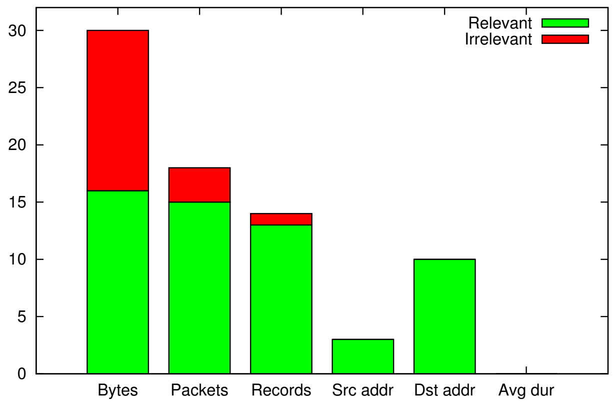


Fig. 9.3: Stacked numbers of relevant and irrelevant alarms of Shewhart control chart applied to ES residuals (ICMP traffic)

ministrator valuable insights in the current state of the network. Whether anomalies found in the byte and packet counts are relevant depends on the ICMP message type. We assume that the network administrator is interested in the detection of high rate pings resembling flooding attacks.

Increased numbers of records and distinct destination IP addresses are caused by ICMP network scans. Although such scans do not represent an instantaneous security threat, they are often followed by further suspicious activities, such as connection attempts to specific TCP ports on the scanned hosts. An anomalously large number of source IP addresses is typically caused by ICMP destination unreachable messages returned from many IP addresses to a single host. This pattern occurs during TCP or UDP scans which are often performed by worms in order to find new victims. In this case, the destination of the ICMP messages leads directly to the worm infected host.

9.2.3 Multi-Metric Analysis of ICMP Traffic

In Section 8.4, the T^2 and T_H^2 control charts of batch-mode PCA achieved best results in comparison with other multi-metric residual generation and change detection methods. Now, we apply these control charts to the traffic measurement time series of ICMP traffic.

As before, we calculate the sample mean and sample covariances from the first two weeks of the measurement data. Table 9.3 shows that the correlation matrix is different from the one calculated over the entire measurement time (see Table 9.1). For example, the correlation between the numbers of flow records and source IP addresses as well as between the numbers of source and destination IP addresses is much larger. We can explain these differences by a smaller number of anomalous values in the training data.

Based on these estimates, the original variables are standardized and

Tab. 9.3: Sample correlation matrix of ICMP traffic (first two weeks)

Bytes	Packets	Records	Src addr	Dst addr	Avg dur
1.0000	0.4331	0.0400	0.0372	0.0117	0.0055
0.4331	1.0000	0.1712	0.1755	0.0732	0.0344
0.0400	0.1712	1.0000	0.9032	0.7344	−0.3727
0.0372	0.1755	0.9032	1.0000	0.6253	−0.3284
0.0117	0.0732	0.7344	0.6253	1.0000	−0.4450
0.0055	0.0344	−0.3727	−0.3284	−0.4450	1.0000

Tab. 9.4: Eigenvalues and w-vectors (ICMP traffic only)

i	1	2	3	4	5	6
Eigenvalue	2.7933	1.4285	0.7879	0.5376	0.3703	0.08214
Bytes	0.0350	0.5787	0.3818	−0.8630	0.0857	−0.0078
Packets	0.0806	0.5748	−0.1195	0.9516	−0.1769	−0.0244
Records	0.3369	−0.0166	−0.2822	−0.1251	0.2958	2.6539
Src addr	0.3214	−0.0033	−0.3514	−0.1102	0.7730	−2.1679
Dst addr	0.3038	−0.0885	0.0380	−0.1609	−1.3560	−0.6506
Avg dur	−0.2026	0.1627	−0.9510	−0.3947	−0.3710	−0.0125

transformed into y-scores. Table 9.4 lists the eigenvalues and eigenvectors (scaled as w-vectors). As shown in the autocorrelation plots in Figure 9.2, daily variation mainly effects the record count and the numbers of source and destination IP addresses. The first PC models this common seasonal variation and gives high loadings to these three metrics. The second PC covers the correlation between the byte and packet counts.

The scree plot of the eigenvalues is displayed in Figure 9.4. As there is no clear knee in the curve, it is difficult to decide on an appropriate number of retained PCs. Therefore, we apply control charts to both Hotelling's T^2 statistic and Hawkins' T_H^2 statistic in order to compare the results. The T_H^2 statistic covers PC4, PC5, and PC6. We use a constant upper control limit of 200 and obtain 37 alarms for the T_H^2 control chart. The T^2 control chart triggers five additional alarms, reaching 42 alarms in total.

The T^2 control chart is depicted in Figure 9.5. We do not show the T_H^2 control chart bacause it looks very similar. For both control charts, all but one alarm are classified as relevant. The only irrelevant alarm resembles an artifact of the measurement data to time series conversion since the numbers of packets and bytes suddenly double in one time interval during of a steady flow of echo messages.

The causes of all detected anomalies are listed in Table 9.5. Most of them go back to ICMP network scans. All scans found in the preceding

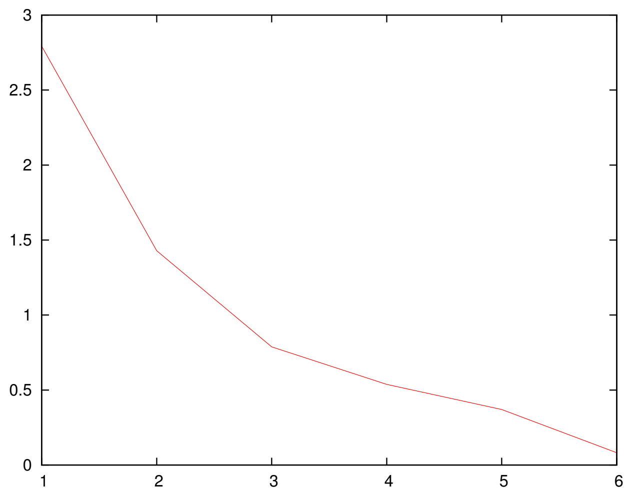


Fig. 9.4: Scree plot (ICMP traffic only)

section with single-metric anomaly detection (see Table 9.2) are detected in the T^2 and T^2_H control charts as well. In addition, a lot of new ICMP network scans are recognized. On the other hand, anomalies which are related to large numbers of bytes or packets are not found. Apparently, these anomalies are not significant enough to cause a peak in the T^2 and T^2_H statistics exceeding the given control limit.

As already concluded in Section 8.4.1, the difference between the T^2 and T^2_H control chart is small. In summary, analyzing ICMP traffic with batch-mode PCA allows detecting a large number of relevant alarms while the number of irrelevant alarms is almost zero.

9.3 Anomaly Detection in SMB Traffic

In Chapter 8, we detected several network scans directed to TCP port 445, which is the well-known port of the Server Message Block (SMB) protocol. Since Windows 2000, SMB is used by Microsoft for file and printer sharing in local area networks. SMB should not be used outside the local network because it does not use encryption and relies on broadcast messages for service discovery. In addition, vulnerabilities in this service can be exploited by worms to infect unprotected computers in the network. A prominent example is the Sasser worm which has been spreading over the Internet since 2004. Therefore, SMB over WAN (Wide Area Network) adapters is disabled by default in recent versions of the Windows operation system. In addition, SMB traffic is usually blocked by local firewalls for security reasons. Nevertheless, there are still a lot of unprotected hosts with older versions of Windows which expose an open SMB port to potential attackers without being protected by a firewall.

Tab. 9.5: Anomalies detected by T^2 and T_H^2 control charts (ICMP traffic only)

Time	Cause	T^2	T_H^2
29/09 10:40 - 29/09 11:20	High rate ping	X	X
30/09 11:05 - 30/09 11:05	ICMP scan, followed by TCP 1433	X	X
19/10 12:20 - 19/10 12:20	ICMP scan, followed by TCP 3389/1433	X	X
24/10 13:20 - 24/10 13:20	ICMP scan, followed by TCP 3389	X	X
25/10 05:35 - 25/10 05:40	ICMP scan, followed by TCP 80/8080/3128	X	X
25/10 06:15 - 25/10 06:15	ICMP scan, followed by TCP 3389	X	X
31/10 22:35 - 31/10 23:55	Low rate pings, measurement artefact?	X	X
02/11 13:25 - 02/11 14:20	Ping flood	X	X
02/11 15:50 - 02/11 15:50	ICMP scan, followed by TCP 5110	X	X
04/11 22:55 - 04/11 22:55	Port unreachable (DNS)	X	X
04/11 23:10 - 04/11 23:10	Port unreachable (DNS)	X	X
05/11 21:10 - 05/11 21:15	ICMP scan, followed by TCP 8028	X	X
06/11 17:55 - 06/11 17:55	multiple small ICMP network scans	X	
07/11 17:40 - 07/11 17:55	multiple small ICMP network scans	X	X
09/11 16:25 - 09/11 16:45	ICMP scan, followed by TCP 15963	X	X
10/11 10:25 - 10/11 10:25	Redirect from host/network (HTTPS)	X	
11/11 07:50 - 11/11 07:50	Port unreachable (HTTPS)	X	X
13/11 03:20 - 13/11 03:25	Port unreachable (HTTPS)	X	X
13/11 09:05 - 13/11 09:05	Port unreachable (HTTPS)	X	
14/11 07:50 - 14/11 07:55	multiple small ICMP network scans	X	
14/11 10:20 - 14/11 10:20	two ICMP network scans	X	
14/11 16:15 - 14/11 17:10	ICMP network scan	X	X
14/11 18:10 - 14/11 19:15	ICMP network scan	X	X
14/11 19:25 - 14/11 19:40	ICMP network scan	X	X
15/11 07:40 - 15/11 07:45	ICMP network scan	X	X
15/11 09:50 - 15/11 10:00	ICMP network scan	X	X
15/11 10:00 - 15/11 10:50	ICMP network scan	X	X
15/11 11:40 - 15/11 12:30	ICMP network scan	X	X
15/11 14:35 - 15/11 14:50	ICMP network scan	X	X
15/11 15:55 - 15/11 16:30	ICMP network scan	X	X
15/11 16:35 - 15/11 16:40	ICMP network scan	X	X
15/11 17:00 - 15/11 17:40	ICMP network scan	X	X
15/11 17:55 - 15/11 18:10	ICMP network scan	X	X
15/11 19:00 - 15/11 19:20	ICMP network scan	X	X
16/11 11:05 - 16/11 11:50	ICMP network scan	X	X
16/11 14:00 - 16/11 15:10	ICMP network scan	X	X
16/11 16:00 - 16/11 16:10	two ICMP network scans	X	X
16/11 16:20 - 16/11 17:10	ICMP network scan	X	X
16/11 18:45 - 16/11 18:55	ICMP network scan	X	X
17/11 07:45 - 17/11 07:50	ICMP network scan	X	X

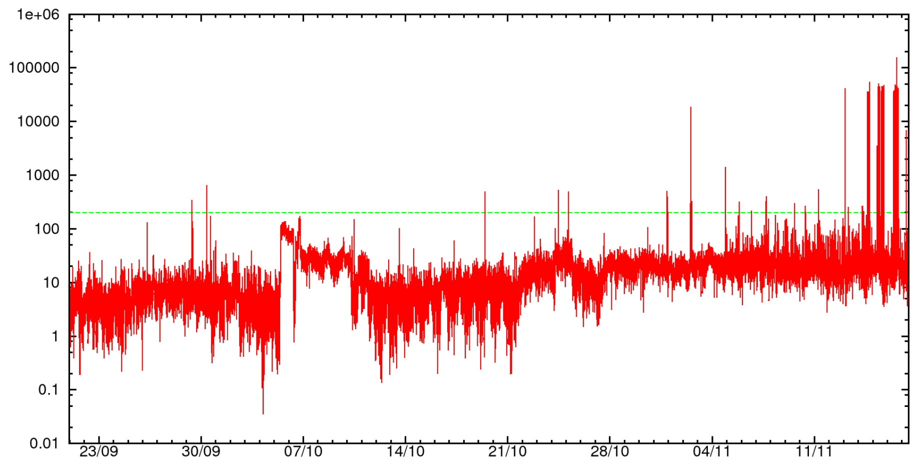


Fig. 9.5: T^2 control chart (ICMP traffic only)

9.3.1 SMB Traffic Characteristics

For our analysis, we consider time series of TCP traffic to and from port 445. Figure 9.6 displays the time series of the different metrics. As can be seen, there is very few SMB traffic most of the time. The sample autocorrelation functions (not shown here) of the measurement variables confirm the visual impression of the time series plots that there is not any significant seasonal variation. All metrics except the average flow duration are significantly correlated to each other. If calculated for the entire measurement time, the correlation coefficients between any two metrics of the number of bytes, packets, flow records, distinct destination IP addresses, and distinct source port numbers are larger than 0.75. As we will see in the following subsections, SMB traffic is dominated by occasional network scans of different intensity, which explains the high correlation between these metrics.

9.3.2 Single-Metric Analysis of SMB Traffic

As before, we apply the Shewhart control chart with EWMS-based adaptive control limits ($\rho = 0.01$) to the residual time series obtained with exponential smoothing ($\alpha = 0.5$). We set the control limits to $\pm 10\hat{\sigma}$ in order to get a reasonably small number of alarms. For all eight metrics, we obtain 198 alarms in 84 different time intervals.

Regarding the causes of the alarms, two types of anomalies are detected very frequently: SMB network scans and repeatedly failed login attempts. For example, the long lasting anomaly between October 5 and October 9, where we observe large values in most of the metrics, is the result of a single host scanning large IP address ranges at relatively high rate. This scan has an effect on the ICMP traffic: In the same period of time, the number of

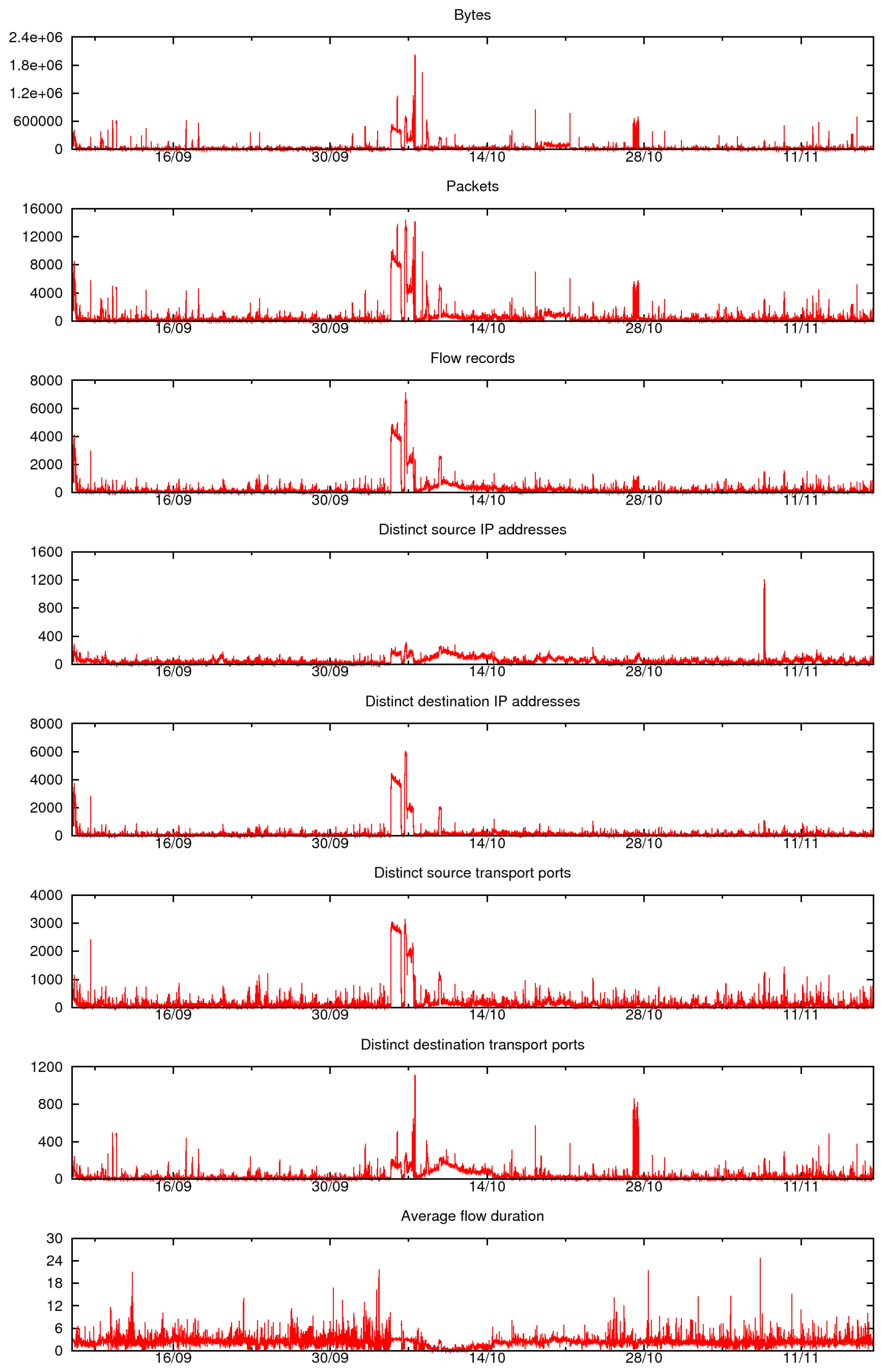


Fig. 9.6: Time series of SMB traffic

flow records and source IP addresses as well as the T^2 statistic of the ICMP traffic are slightly increased (see Figures 9.1 and 9.5, respectively). These increases are caused by ICMP destination unreachable messages which are returned to the scanning host because a large number of TCP packets cannot be delivered to the scanned destinations. The scanning host is very probably infected by a worm trying to connect to randomly chosen destinations.

Apart from the extreme network scan from October 5 to October 9, 41 shorter periods of scan traffic are detected. These scans originate from IP addresses of remote subnets, hence we probably observe a part of the scan traffic only, namely those packets which are directed to a subnets of the ISP network. This assumption is fortified by the fact that the scanned IP addresses share only four different /16 prefixes. In contrast, the long scan in October is caused by a host within the ISP network, hence we observe a large part of the traffic to many different destination IP addresses. 15 of the short scans involve 800 to 900 destination IP addresses, another 10 scans between 300 and 600 destination addresses. In all cases, return traffic is observed from less than 150 hosts. The other hosts probably do not exist or port 445 is blocked.

A failed SMB login attempt results in a bidirectional flow with less than 10 packets in each direction. In the case of a successful login, the TCP connection persists for a longer period of time and results in flows with a larger number of packets and bytes. Hence, a large number of flows with similarly small numbers of packets exchanged between two IP addresses are a sign of repeatedly failed SMB login attempts. Such traffic is likely caused by an automated tool performing brute-force password guessing. Another explanation is that a worm tries to infect the target host with different exploits. However, we assume that password guessing is the more probable reason.

Large numbers of failed login attempts represent the second most frequent anomaly pattern detected in the SMB traffic, after SMB network scans. Sometimes, we only see one direction of the traffic, probably due to asymmetric routing. The IP addresses of the affected SMB servers are in the subnets of the ISP network whereas the IP addresses of the clients are spread over the entire address range.

A couple of network scans trigger an alarm in the average flow duration. In this case, some of the scan packets are repeated after a certain amount of time. Having identical flow key values, such duplicate packets are accounted in a single flow records, causing a long flow duration. Furthermore, three anomalies in this metric go back to flows with only a few packets but a large time difference between the first and last packet of the flow, such as 55 seconds or even 103 seconds. Here again, we do not know if these packets actually belong to the same TCP connection or to different TCP connections with identical IP quintuples.

On November 7 and 9, two anomalies are caused by many small flows

Tab. 9.6: Type of detected anomalies per metric

Metric	Network scan	Password guessing	Long duration	Small flows	Large flow	Total
Bytes	11	28	0	2	2	43
Packets	12	18	0	2	0	32
Records	18	5	0	2	0	25
Src addr	10	0	0	1	0	11
Dst addr	22	0	0	1	0	23
Src port	14	1	0	1	0	16
Dst port	18	18	0	2	0	38
Avg dur	8	0	3	0	0	11
All	43	28	3	2	2	78

containing no more than four packets with various source and destination IP addresses. Both anomalies last for more than two hours. The pattern differs from the fingerprints of network scans and failed login attempts which are characterized by flows originating from or terminating at a specific IP address. The sudden occurrence of all these small flows suggests that they are somehow related to each other. However, as we do not know the reason for this traffic, we classify the alarms as irrelevant.

Finally, there are two anomalies caused by flows with long duration and a large number of packets. Only in these two cases, there seems to be regular SMB activity. However, the number of exchanged bytes is quite low and does not indicate a large data transfer.

Table 9.6 shows how many anomalies of a specific type are detected in the time series of different metrics and in all metrics together. The last row indicates the total number of anomalies of a given type. As the same anomaly may be detected in different metrics, the total number of anomalies is not equal to the sum of anomalies found in the individual metrics.

One would expect that network scans were principally detected in the number of distinct destination IP addresses. In fact, most scans with more than 500 destination IP addresses are detected in this metric. However, scans with few destinations are often detected in one of the other metrics only. As already mentioned, delayed duplicate packets which are accounted in the same flow record are the reason for scans being detected in the average flow duration.

Password guessing (i.e., a large numbers of failed login attempts) is detected in the number of bytes. Two thirds of these anomalies are also detected in the number of packets or the number of distinct destination ports.

Flows with few packets but long duration trigger an alarm in the average flow duration. Anomalies caused by many small flows become visible in the number of bytes, packets, flows, and destination ports. Finally, anomalies which are provoked by individual long flows are detectable in the byte counts.

For detecting the described anomalies, it is not necessary to monitor the number of distinct source ports because all alarms detected in this metric are found in at least one of the other metrics as well. All other metrics should be kept, otherwise we would miss at least one of the anomalies.

Our results suggest that monitoring TCP traffic to and from port 445 is worthwhile. Almost any traffic which exceeds the regular background noise of sporadic packets can be related to a suspicious traffic pattern. The analysis of the involved IP addresses allows us to identify potentially infected hosts and hosts which are prone to infections because they open the SMB service to the public Internet.

9.3.3 Multi-Metric Analysis of SMB Traffic

For multi-metric anomaly detection, we transform the standardized measurement variables into y-scores using batch-mode PCA. As before, we use the first two weeks of data for estimating the mean and covariances. Thereafter, we calculate the T^2 statistic as well as the T_H^2 statistic of PC4 to PC8 and apply control charts with constant control levels to the resulting time series. As a result, we detect several additional alarms which are all related to the same causes as explained in the previous subsection. On the other hand, some of the alarms detected by the single-metric approach are not found by the T^2 and T_H^2 control charts.

We omit a further discussion of the detection results. In summary, T^2 and T_H^2 control charts applied to SMB traffic provide good detection results. The detected alarms complement those obtained with single-metric control charts.

9.4 Discussion of Results

In this chapter, we applied single-metric and multi-metric residual generation and change detection methods to specific subsets of the analyzed flow dataset, namely to ICMP and SMB traffic. We chose these two protocols because normal users and applications are not expected to use them in an ISP network. Hence, any anomalous changes in the corresponding parts of traffic are suspicious from the outset.

Our evaluation results confirm that monitoring ICMP traffic is very interesting for the network administrator. One the one hand, ICMP is often used for network scans. On the other hand, ICMP messages also appear in large numbers during TCP or UDP scans in order to report unreachable destinations. Last but not least, increased ICMP traffic may occur in the case of network or server problems.

Traffic measured in an ISP network should not contain any SMB traffic because this protocol is to be used in local networks only. Indeed, we identified only two SMB connections which resembled legitimate traffic. In all other time intervals with increased SMB traffic, we found suspicious patterns of SMB network scans or large numbers of failed login attempts, probably going back to password guessing.

We expect that the focused analysis of other parts of traffic would also produce interesting results. For example, we found multiple incidents of SSH password guessing in the overall IP traffic although SSH represents a small proportion of the traffic only. Hence, examining SSH traffic separately from the remaining traffic will very probably reveal more of these incidents. In contrast to ICMP and SMB, however, we must not consider all SSH traffic as suspicious because there are legitimate use cases for this protocol, such as remote shell access or file transfers. Therefore, anomaly detection may result in many irrelevant alarms unless we restrict it to metrics which are mainly sensitive to password guessing, such as the numbers of distinct source and destination ports.

Alternatively to selecting specific parts of traffic, we can remove the traffic of applications which are known to cause a lot of irrelevant anomalies in the measurement time series. For example, it could be worth analyzing UDP traffic without DNS in order to reduce the number of irrelevant alarms. DNS traffic can be very reliably identified and filtered out using the well-known port 53.

Regarding the detection results of Chapter 8 and this chapter, we see that the majority of relevant alarms goes back to scanning activities and password guessing. For the investigation of anomaly causes, it would be very helpful if these kinds of anomalies were identified without requiring any manual examination of the original flow data. For this purpose, we have developed identification algorithms which automate the search for patterns of scans and password guessing. These algorithms proved to be very useful for our evaluation of the anomaly detection results. A presentation of the identification algorithms follows in the next chapter.

10. IDENTIFICATION OF ANOMALY CAUSES

10.1 Introduction

Chapters 8 and 9 were dedicated to the evaluation and comparison of different anomaly detection methods. One aspect of the evaluation concerned the relevance of the detected anomalies for the network administrator. Therefore, we identified the likely causes of the detected anomalies by examining the original flow data.

The investigation of an anomaly usually starts with the determination of **'heavy hitters'**, which are hosts generating more traffic volume or contacting a larger number of destinations than the other hosts. For this purpose, we determine the source IP addresses with the largest amount of outbound traffic measured in bytes, packets, or flow records, or the largest numbers of distinct destination IP addresses or ports (sometimes called 'fan-out') observed in the time interval of the anomaly to be inspected. Heavy hitters generating large traffic volume are often popular servers sending data to their clients. In this case, the local port can help to identify the application or service if a well-known port number is used, such as TCP port 80 for HTTP (i.e., web traffic) or UDP port 53 for DNS (Domain Name Service). Alternatively, the heavy hitter could be a client which is uploading large amount of data to a server or to other peers in a peer-to-peer (P2P) network. A heavy hitter which sends packets to a lot of destination IP addresses or destination ports looks like a **'scanner'**, that is a host performing a network or port scan, respectively.

The identification of heavy hitters provides the basis for an initial suspicion regarding the cause of an anomaly. In order to validate or falsify our assumption, we need additional information about the characteristic of the traffic. Furthermore, it is useful to have a close look at the reverse traffic, if there is any. Last but not least, we need to verify if the identified flows are actually responsible for the detected anomaly. If the effect on the measurement variables is very small, or if the identified cause has already existed before the alarm, we might be off the track and should look for an alternative explanation of the anomaly.

The identification of anomaly causes is not only required for our evaluation but also for the practical deployment of traffic anomaly detection methods in general. The mere information about an anomaly is not very helpful for the network administrator because he needs much more details about what is happening in the network in order to assess the situation and react accordingly. It is out of the question that the network administrator identifies the anomaly causes manually because this work can be very time consuming. Instead, additional information and a precise description of the traffic which causes the detected anomaly should be automatically provided as far as possible.

In this chapter, we present identification algorithms which enable the recognition of three frequent causes of relevant anomalies, namely network scans, port scans, and large numbers of failed login attempts which are likely caused by brute-force password guessing. Given the time interval in which an anomaly has been detected, these algorithms search the flow records for characteristic patterns for these kinds of incidents.

In the past, a couple of methods for identifying harmful traffic patterns in packet or flow data have been presented. We give an overview on this related work in the next section. A major difference to our approach is that most of the existing methods maintain data structures which are specific to the traffic patterns to be identified. In contrast, our identification algorithms are based on a flow database storing the original flow records. From this database, the necessary information is retrieved on demand using appropriate SQL (Structured Query Language) queries. We do not keep any additional data structures or states. In many networks, flow repositories are already used for other purposes, such as traffic monitoring or accounting. Hence, the effort to implement and deploy our identification algorithms is low. The table structure of the flow database as well as other prerequisites of the algorithms are described in Section 10.3.

In Sections 10.4 and 10.5, we present two algorithms for the detection of network and port scans. Section 10.6 explains how to identify clients which establish a large number of short connections to the same server, which is a characteristic traffic pattern for many failed login attempts. In Section 10.7, we discuss the resources required to perform the database queries of the identification algorithms before Section 10.8 concludes this chapter.

10.2 Existing Approaches for Identifying Harmful Traffic Patterns

In this section, we give an overview on related work dealing with the detection and identification of specific patterns of harmful traffic. The overview focuses on methods which have been explicitly proposed for the analysis of flow data, and on methods which originally work on packet data but could be applied to flow data as well. Research in this area has principally aimed at the detection of scans, worm propagation, and flooding attacks. On the other hand, we are not aware of any methods considering the flow-based identification of failed login attempts.

In a presentation of the flow analysis tool suite *SiLK* [GCD$^+$04], Gates et al. give several examples how these tools can be used for security analysis. Particularly, the authors demonstrate how to identify network scans and SYN flooding attacks. In principle, the analysis is based on the recognition of heavy hitters as sketched in the introduction of this chapter. As an alternative method to detect network scans with *SiLK*, McHugh suggests to determine all source IP addresses which appear in the flow records [McH04]. The resulting list of IP addresses represents the active hosts in the network under the condition that address spoofing can be excluded and asymmetric routing does not inhibit the monitoring of bidirectional traffic. Hosts sending packets to a large number of IP addresses which do not belong to this list are classified as potential scanners.

SiLK uses a file-based flow repository and implements a set of tools which enable specific queries on the stored flow records, including the application of filters and certain types of heavy hitter identification. Similar open-source tool suites with equivalent functionality are *flow-tools* [FR00] and *nfdump* [NfD10]. However, the functionality offered by *SiLK*, *flow-tools*, and *nfdump* is not sufficient to realize the identification algorithms proposed in this chapter. In order to implement our algorithms upon the file-based flow repositories, we would have to write additional query functions. Therefore, our current implementation is based on a flow database which allows us to perform very flexible and complex queries on the flow records using regular SQL statements. More information about the flow database is given in the next section.

Binkley et al. propose the analysis of host behavior with help of several statistics of inbound and outbound traffic, including counters for the number of TCP packets with specific TCP flags (SYN, SYN/ACK, FIN, and RST) [BMG05]. Among others, the authors define a profile for hosts which are likely infected by a worm. Although the host classification method has been originally conceived for packet data analysis, it should work in a similar way with flow records as its input. A necessary condition is, however, that the TCP control bits field is included in each flow record, containing the disjunction of all TCP flags observed in the packets of the flow.

Leckie and Ramamohanarao [LR02] propose an algorithm for network scan detection which is based on the assumption that scanners choose their targets randomly with equal probability. Regarding the entire traffic, though, the frequency distribution of destination IP addresses does not resemble a uniform distribution. As a consequence, a scanner is expected to send many packets to destinations which rarely occur in the overall traffic. On the other hand, a normally behaving host is expected to mainly communicate with destinations which are frequently accessed. Since it is not useful to classify a host as a scanner just because it has sent a single packet to a rare destination, the number of distinct destinations is also taken into consideration. The authors assume that the number of destination IP addresses contacted by a normal host is Poisson distributed whereas a scanner may send packets to an arbitrary number of destinations which is uniformly distributed and limited only by the size of address space. The algorithm needs to maintain a list for every host which contains the contacted destination IP addresses. Based on this information, it calculates the probabilities of normal and scanning behavior for every host and classifies them according to the maximum likelihood.

In the case of TCP, a useful statistic for detecting scans is the number of failed connection attempts. Under normal conditions, most TCP connections initiated by a host will be successful. In contrast, the number of failed connection attempts will be high if the host is performing a TCP network or port scan. The scan detection algorithm *Threshold Random Work* (TRW) by Jung et al. records the number of successful and failed connection attempts for a given host and performs sequential hypothesis testing until the host can be classified as benign or as a scanner [JPBB04]. This algorithm has been integrated into the network intrusion detection system *Bro* [Pax99]. If the TCP control bits field is available in the flow records, we can distinguish successful and failed TCP connection and thus apply the TRW algorithm. In fact, the TRW algorithms has been successfully implemented as flow analysis module for *VERMONT* [VER10] (see Section 2.4.2).

In his master's thesis, Malmedal proposes a flow-based scan detection method which determines heavy hitters with the largest numbers of failed connection attempts [Mal05]. As before, the TCP control bits field is needed to identify failed connection attempts. Similar to our approach, flow records are stored in a database on which queries are performed using SQL statements.

An important aspect is the validity of the assumed traffic patterns. For example, if the observed traffic is compared to the expected behavior of a client host, the corresponding method may fail in the case of a server which shows very different inbound and outbound traffic characteristics. In particular, the number of destination IP addresses will be very large even under normal conditions. We assume that most of the existing approaches result in an increased number of false positives if the behavior of regular hosts varies a lot. In this context, a false positive is the identification of an incident which has not occurred. Our identification algorithms reduce the risk of false positives by considering multiple aspects of the traffic, including the properties of reverse traffic and the number of packets per flow.

10.3 Prerequisites

The identification algorithms presented in the following sections require that the original flow records of the inspected time interval are available. The necessary storage capacity mainly depends on the number of flow records. Furthermore, the storage system must provide an interface which enables

Tab. 10.1: Flow table layout

Flow key fields

Column name	Data type	Description
srcIP	uint32	Source IP address
dstIP	uint32	Destination IP address
srcPort	uint16	Source port or 0 if not available
dstPort	uint16	Destination port or ICMP type/code
proto	uint8	Transport protocol

Non-key fields

Column name	Data type	Description
bytes	uint64	Number of bytes
pkts	uint64	Number of packets
firstSwitched	uint32	Flow start in seconds (unix time)
lastSwitched	uint32	Flow end in seconds (unix time)

the identification algorithms to perform different kinds of queries on the flow records.

Database systems offer powerful data processing and querying functions and are optimized for handling large amounts of data. Hence, we decided to store the flow records in a database in order to profit from the existing functionality. The advantage of this approach is that the identification algorithms can be realized with small sequences of SQL statements. Alternatively, the algorithms could be implemented for file-based flow repositories, such as those generated by *SiLK*, *flow-tools*, or *nfdump*. As mentioned in Section 10.2, however, the current functionality offered by these tool suites is not sufficient and would have to be extended in order to support the required queries.

Table 10.1 shows the flow key and non-key fields which are stored for each flow record. The flow key fields correspond to the common IP quintuple, the non-key fields to the timestamps and statistics which are typically exported by the flow metering process. If ICMP is transport protocol, the field of the destination port number is overloaded with the ICMP message type and code, which is common practice in flow records of *Cisco NetFlow*.

The TCP control bits field is not included in the table although the information encoded in this field would allow us to assess very easily if a TCP connection has been initiated, successfully established, rejected, or terminated. Nevertheless, our identification algorithms currently do not make use of TCP control bits because the corresponding field is not always available in the *NetFlow* records exported by Cisco routers.

As mentioned in Section 7.6, we use *MySQL* [MyS10] for our flow database. Flow records are stored in flow tables, each containing 30 minutes of flow data. We decided to start a new flow table every half an hour in order to limit the size of a single table. The inspection of an anomaly typically focuses on a time span of a few minutes, which means that flow records stored in one or two neighboring tables are concerned. In the later case, the two tables can be virtually merged in one table using the MERGE storage engine of *MySQL*.

Although splitting the flow data into time bins of exactly 30 minutes is arbitrary, it generally makes sense not to store all flow records in one large table. Firstly, queries on large tables are slow unless appropriate table indexes exist. Secondly, we can easily drop old flow tables if the stored flow data is not needed any more. This facilitates the implementation of a round robin database (RRD) which only keeps the most recent flow data. If a single table was used, the database system would have to search the rows to be removed, which is a much more complex operation.

Using an index for appropriate columns can reduce the query time significantly. On the other hand, building indexes requires additional computational resources as well as additional memory to store the index files. In order to keep the index up-to-date, it has to be updated with every new row, which slows down the insertion of flow records into the database. As this may cause a bottleneck if new flow records are received at a very high rate, we do not use any indexes in our implementation.

10.4 Network Scans

10.4.1 Network Scan Characteristics

Network scan can be performed using ICMP, TCP, or UDP as transport protocol. The purpose of an ICMP network scan is to find active hosts in the network. Network scans based on TCP and UDP are used to check if a specific port is open on the target hosts.

In the case of an ICMP network scan, the scanner emits ICMP echo request messages to the target IP addresses. If the IP address is in use, the corresponding host will answer the echo request with an ICMP echo reply message. Hence, the scanner knows for sure that the IP address belongs to an existing and reachable host if it receives a reply message. However, both echo request and reply messages may be blocked by firewalls. Therefore, a missing reply messages does not necessarily mean that the target IP address is indeed unused.

During a TCP network scan, the scanner emits TCP SYN packets with the same destination port to different target IP addresses. If a target host exists and if the corresponding port is open, the connection is usually accepted by returning a SYN/ACK packet. In this case, the scanner knows that the port on the target host is open and accessible. If the scanner receives a RST/ACK packet instead of a SYN/ACK packet, the target host has rejected the connection attempt, typically because the given port is closed. Quite often, however, the scanner does not get any response at all. This is because many hosts are protected by firewalls blocking incoming TCP packets which do not belong to an existing connection and which are directed to ports that are not supposed to offer any service to external users.

TCP network scans may also use packets with other TCP flags. In the past, scans with FIN packets or packets with invalid flag combinations, such as SYN/FIN and FIN/PSH/URG (called 'XMAS scan'), have been used to pass firewalls with incorrectly implemented packet filters. These scans result in the return of RST/ACK packets if the given port is open. ACK packets with invalid sequence numbers, which can also be used for scans, are always answered with RST/ACK packets, independently of whether the port is open or closed. Hence, ACK scans can be used to find active hosts, yet they do not provide any information about open ports.

A UDP packet arriving at a closed port typically results in an ICMP port unreachable message returned to the sender. If the port is open, the reaction depends on the service or application bound to this port. Only if the UDP packet is answered with reverse traffic of any kind, the scanner is able to identify an open UDP port. Due to the limited utility, it is not surprising that we did not find any anomaly related to UDP scans in the analyzed dataset.

Regardless of whether the scanner uses ICMP, TCP, or UDP packets, ICMP host unreachable messages should be returned by routers in the network if the target IP address is not assigned to any host. In this case, the scanner knows for sure that the target does not exist. If a packet is blocked due to a packet filter in a firewall, an ICMP destination unreachable messages with code 'communication administratively prohibited' may be returned. In fact, some of the ICMP anomalies found in Section 9.2 are related to this kind of ICMP messages.

In Chapters 8 and 9, TCP and ICMP network scans turned out to be frequent causes of anomalies. After a scanner had hit an open TCP port, we often observed a complete TCP connection establishment and the exchange of additional packets. Sometimes, the scanning host opened additional TCP connections to the same target host to infect or break into the system. For example, we detected multiple network scans to ports 22 (SSH) and 445 (SMB) where a small proportion of the scanned hosts accepted the TCP connections. As a consequence, we observed multiple short TCP connections to these hosts resembling the traffic pattern of failed login attempts. In Section 10.6, we present an algorithm for identifying this kind of pattern.

Sometimes, we observed that a TCP network scan targeted multiple ports in parallel. As an example, we detected simultaneous networks scans to ports 1433 and 3389 or ports 80, 8080, and 3128 in Section 9.2.2. We also observed that ICMP network scans were often followed by TCP connection attempts to specific services. This means that some scanners first scanned a wide range of IP addresses using ICMP echo request messages before continuing with a TCP network scan to those hosts that proofed to be reachable via ICMP.

In a recent study on the evolution of scanning in the past years, Allman et al. report that scans have become very frequent since 2000 [APT07]. In the dataset analyzed by the authors, the majority of all TCP connections observed in 2003 and later go back to scans. Network scans are much more frequent than port scans. On average, around 500 hosts are scanned during a network scan, yet a few scanners target several thousand IP addresses. These observations correspond to our traffic analysis results. Unfortunately, the paper only deals with TCP scans although the evolution of ICMP scans would have been very interesting as well.

10.4.2 Algorithm for Network Scan Identification

As we have seen in Chapters 8 and 9, network scans cause large numbers of distinct destination IP addresses. Further metrics which are sensitive to network scans are the number of records and, in the case of TCP network scans, the number of distinct source ports. Hence, an anomaly detected in at least one of these metrics is a hint of a network scan taking place in the corresponding time interval.

A network scan is characterized by many flows originating from the same source IP address and directed to different destination IP addresses. Therefore, we define a threshold D_{min} for the minimum number of target hosts to be contacted by a scanner within the given time interval. D_{min} divided by the length of the time interval results in the lowest scan rate that can be detected if the scan lasts for the entire interval. For an interval length of 5 minutes, $D_{min} = 50$ is a good compromise, corresponding to a minimum sustained scan rate of 10 hosts per minute.

Flows involved in a network scan are typically composed of only one packet. Sometimes, however, we observed flows containing more than one packet, maybe because the scanner sent duplicate packets. On the other hand, flows with large numbers of packets very unlikely belong to a scan. Therefore, it is save to ignore flows with more than P_{max} packets, P_{max} being a small positive number. In practice, $P_{max} = 3$ is an appropriate value as it filters out all established TCP connections and keeps flows containing occasional retransmissions of scan packets. This value also works fine for ICMP network scans.

A third characteristic of a network scan is that reverse traffic is only observed for a small proportion of the target hosts because many scanned hosts do not exist. More generally, the number of hosts sending packets to the scanner is much lower than the number of scanned hosts. Otherwise, the scanner is either extremely successful, which is very unlikely, or the traffic is not caused by a scan but belongs to an application with similar characteristics.

```
Input: Flow records of the given time interval, expected maximum number
       of packets in a scanning flow $P_{max}$, minimum number of scanned IP
       addresses ($D_{min}$)
Output: ($srcIp, proto, dstPort$) tuple of each identified network scan
        ($dstPort$ omitted if not applicable)
1  From all flows containing at most $P_{max}$ packets, determine all
   ($proto_i, srcIp_i, dstPort_i$) tuples for which the numbers of distinct
   destination IP addresses $\#dstIps_i$ are $\geq D_{min}$;
2  for $i = 1, 2, \ldots$ do
3      if protocol does not use ports (e.g., ICMP, $proto_i = 1$) then
4          From the subset of flows with protocol $proto_i$ and destination IP
           address $srcIp_i$, determine the number of distinct source IP
           addresses $r$;
5          if $r \ll \#dstIps_i$ then
6              Optionally, determine the subset of destination IP addresses of
               flows with source IP address $srcIp_i$ (i.e., the scanned IP
               addresses) and the subset of source IP addresses of flows with
               destination IP address $srcIp_i$ (i.e., the responding hosts) and
               verify if the second subset is included in the first one;
7              Optionally, check if the scanning packets are actually observed
               before the corresponding response packets;
8              Report a network scan with protocol $proto_i$ originating from
               $srcIp_i$;
9          endif
10     else
11         From the subset of flows with protocol $proto_i$, destination IP
           address $srcIp_i$, and source port $dstPort_i$, determine the number of
           distinct source IP addresses $r$;
12         if $r \ll \#dstIps_i$ then
13             Optionally, determine the subset of destination IP addresses of
               flows with source IP address $srcIp_i$ and destination port
               $dstPort_i$ (i.e., the scanned IP addresses) and the subset of
               source IP addresses of flows with destination IP address $srcIp_i$
               and source port $dstPort_i$ (i.e., the responding hosts) and verify
               if the second subset is included in the first one;
14             Optionally, check if the scanning packets are actually observed
               before the corresponding response packets;
15             Report a network scan with protocol $proto_i$ originating from
               $srcIp_i$ to port $dstPort_i$;
16         endif
17     endif
18 endfor
```

Alg. 1: Network scan identification

Based on the above considerations, we developed the identification algorithm shown in Algorithm 1. The first query in line 1 identifies the IP addresses of potential scanners by analyzing the outbound traffic of all hosts. In SQL, the query looks as follows (`<flow_table>` being the placeholder of the table name):

```
SELECT proto, srcIp, dstPort, COUNT(DISTINCT dstIp) AS di
  FROM <flow_table> WHERE pkts<=3
  GROUP BY proto, srcIp, dstPort HAVING di>=50;
```

In order to respond to this query, the database determines the number of distinct destination IP addresses for all flows which contain at most P_{max} packets and share the same triple of protocol, source IP address, and destination port. Triples with less than D_{min} distinct destination IP addresses are discarded, the others are kept for further inspection. The source IP addresses of the kept triples belong to potential scanners. In the case of TCP, the associated destination ports indicate the scanned ports.

As the ICMP message type and code are encoded in the destination port column of ICMP flows, we can verify whether the scanning flows consist of ICMP echo request messages. In the case of other ICMP messages, it is not an ICMP network scan. In Algorithm 1, we omit this check.

The second query determines the number of distinct source IP addresses responding to a potential scanner. The query is different for transport protocols with ports (line 10) and without ports (line 4) . Only if ports are used, we know that the source port of the returned flows must be equal to the scanned port. The response traffic must always be directed to the IP address of the scanner. The query in line 10 corresponds to the following SQL statement:

```
SELECT COUNT(DISTINCT srcIP) FROM <flow_table>
  WHERE proto=<proto> AND srcPort=<scanned_port> AND
  dstIp=<scanner_ip>;
```

`<proto>`, `<scanned_port>`, and `<scanner_ip>` are placeholders for transport protocol, the scanned port, and the IP address of the scanner. In the query of line 4, the condition on the source port is omitted. If the destination port field is overloaded with message type and code in the case of ICMP, we could check if the return flows contain ICMP echo reply messages.

Next, the number of scanned IP addresses is compared to the number or responding IP addresses (lines 7 and 13). If the number of responding IP addresses is significantly smaller, it is very certain that we have found a network scan. For further verification, we could check if the responding IP addresses actually match any of the scanned IP addresses. Furthermore, we could compare the timestamps of two matching flows in order to confirm that the scan packets precede the returned packets. To do so, the available timestamps would need to be accurate enough to determine the order of the flow starts correctly. As it is unlikely that these two additional checks lead to a falsification of the network scan hypothesis, the comparison of the numbers of IP addresses is sufficient in practice.

10.4.3 Limitations

The algorithm assumes that a scanner contacts at least D_{min} different destination IP addresses in the given time interval. If a network scan addresses a smaller number of hosts, the scanner remains unnoticed in the first step of the algorithm. We may also miss a network scan if the scanning speed is very low, or if the scan is performed with distributed hosts, each sending packets to a small number of targets only. The probability to identify slow network scans can be increased by decreasing D_{min} or by applying the algorithm to a larger time interval.

Furthermore, the algorithm assumes that only a small number of scanned hosts answer the scanner. If the scanner is very successful and receives responses from the majority of scanned hosts, the traffic does not correspond to the expected pattern of a network scan. The scanner can evade detection by sending duplicate packets so that more than P_{max} packets are accounted in each flow record.

Finally, the algorithm may misclassify other traffic as a network scan. For example, if asymmetric routing inhibits that we observe both outbound and inbound traffic of a specific host, the resulting traffic flows could match the given pattern.

10.5 Port Scans

10.5.1 Port Scan Characteristics

Port scans can be performed for transport protocols using ports, such as UDP or TCP. In Section 10.4.1, we explained the different cases which may occur if a TCP or UDP packet is emitted during a network scan. The same considerations apply to port scans.

The difference between network scans and port scans is that a port scanner sends packets to multiple ports on a single host in order to determine which ports are open. If the target hosts exist and if no firewall blocks parts of the traffic, the host will reply to every TCP SYN packet with a SYN/ACK or RST/ACK packet, depending on whether the port is open or closed. Hence, in contrast to network scans, the number of reverse flows is more or less equal to the number of scanning flows. In the analyzed dataset, however, we only observed few reverse flows during port scans, probably due to firewalls.

In the case of UDP, the target host returns an ICMP port unreachable messages if a UDP packet is received on a closed port. As ICMP flows are only distinguished by the message type and code, there will be a single ICMP flow returned to the scanner.

In the analyzed dataset, we only found two anomalies caused by TCP port scans. A possible explanation is that large port scans are not very interesting for worms and hackers because the ports of the services to be attacked are typically known in advance. Hence, it is sufficient to send packets to specific ports only.

The first port scan contained packets directed to more than 34,000 different ports. Reverse traffic was observed for two ports only. The second scan was more selective and checked around 6,200 ports. There was no reverse traffic in the dataset, maybe due to asymmetric routing. In both cases, the number of distinct source ports used by the scanner was less than 50, which means that the same local port was used for multiple destination ports. This behavior suggests that the scans were not performed with normal TCP sockets because these would use a new source port for every connection attempt.

10.5.2 Algorithm for Port Scan Identification

In Section 8.3.3, port scans have been detected because of anomalously large numbers of flows or distinct destination ports. The expected pattern in the flow records consists of several flows which are sent from the same source IP address to the same destination IP address. Every flow is directed to another destination port. Typically, each flow contains one packet only. Due to duplicates, however, there might be more than one packet in a flow. Therefore, we consider flows with up to $P_{max} = 3$ packets, just like in the case of network scans. Furthermore, we assume that a scanner sends packets to at least $D_{min} = 50$ different ports in the given time interval.

In the case of TCP, we can expect that the number of reverse flows is not larger than the number of scanning flows, and that the number of distinct source ports in the reverse flows is not larger than the number of scanned ports. In the case of UDP, the target host may return UDP packets if the port is open. As some of the scanned ports are very probably closed, we can expect a flow of ICMP port unreachable messages unless these messages are blocked by a firewall. Again, the number of reverse flows is not larger than the number of scanning flows.

It is useful to have a look at the source ports of the scanning flows. A large number of distinct source ports fortifies the assumption that the observed traffic is a port scan because using a new ephemeral port for every TCP connection attempt corresponds to the behavior of a regular TCP client. In our flow data, however, we saw that the scanner used the same source port for a large number of destination ports. Hence, a small number of source ports must not lead to the rejection of the port scan hypothesis. At

```
  Input: Flow records of the given time interval, expected maximum number
         of packets in a scanning flow $P_{max}$, minimum number of scanned
         ports ($D_{min}$)
  Output: ($srcIp, proto, dstIp$) tuple of each identified port scan
1 From all flows with source port > 1023 and at most $P_{max}$ packets, determine
  the number of flow records $\#records_i$ of all ($proto_i, srcIp_i, dstIp_i$) tuples for
  which the numbers of distinct destination ports $\#dstPorts_i$ are $\geq D_{min}$ ;
2 for $i = 1, 2, \ldots$ do
3     Determine the number of flow records $r$ with protocol $proto_i$,
      destination IP address $srcIp_i$, source IP address $dstIp_i$, and destination
      port > 1023 ;
4     if $r \leq \#records_i$ then
5         Verify if the variation in the average packet length is small for those
          flow records with source IP address $srcIp_i$, destination IP address
          $dstIp_i$, and at most $P_{max}$ packets;
6         Optionally, check if the scanning packets are actually observed
          before the corresponding response packets;
7         Report a network scan with protocol $proto_i$ originating from $srcIp_i$
          and targeting $dstIp_i$;
8     endif
9 endfor
```

Alg. 2: Port scan identification

least, the source port used by the scanner is very likely not a well-known port and therefore larger than 1023. If the source port is a registered port of a common application or service, we should check very carefully if we actually observe regular application traffic instead of a port scan. Section 10.5.3 discusses this problem in more detail.

Taking into account the above considerations, Algorithm 2 finds patterns of port scans. The query in line 1 determines all triples of protocol, source IP address, and destination IP address in the flow records with at most P_{max} packets and source port larger than 1023 (i.e., beyond the range of well-known ports). A triple identifies a potential port scan if the number of distinct destination ports is D_{min} or larger. Using SQL, the query looks as follows:

```
SELECT proto, srcIp, dstIp, COUNT(*),
  COUNT(DISTINCT dstPort) AS dp FROM <flow_table>
  WHERE pkts<=3 AND srcPort>1023
  GROUP BY proto, srcIp, dstIp HAVING dp>=50;
```

The source IP addresses of the resulting triples are the IP addresses of potential scanners.

The second query in line 3 obtains the number of flow records in reverse direction. The corresponding SQL statement is:

```
SELECT COUNT(*) FROM <flow_table> WHERE proto=<proto>
  AND srcIp=<scanned_ip> AND dstIp=<scanner_ip>;
```

`<proto>`, `<scanned_ip>`, and `<scanner_ip>` are placeholders for the transport protocol, the IP address of the scanned host, and the IP address of the scanner, respectively. If the number of flows in reverse direction does not exceed the number of flow records from the scanner to the scanned host, this may be a port scan.

As a further step, we propose to verify that most of the scanning flows have the same average packet length. If TCP SYN packets are used for the scan, the packet lengths will very probably be identical. If UDP is used, we can also expect that the packet lengths are the same. On the other hand, flows with different average packet lengths likely belong to an application or service. Hence, examining the average packet length helps to exclude that we mistake application traffic for a port scan.

Just like in the case of network scan identification, we could compare the start timestamps of the scanning flows and the corresponding reverse flows. Hence, it would be possible to distinguish port scans from client-server traffic with the server being located at the position of the presumed scanner. This analysis, however, requires that the timestamps are very accurate, which is often not the case.

10.5.3 Limitations

Just like the network scan identification algorithm presented in Section 10.4, port scan identification fails if the number of scanned targets (i.e., scanned ports) is smaller than D_{min} in the inspected time interval. According to our experience, we can use relatively small values for D_{min}, for example $D_{min} = 20$, without significantly increasing the number of source IP addresses found in the first query (line 1). This reduces the risk of not detecting a port scan.

Another conceptional problem arises if the scanner uses a well-known port number as source port. However, we assume that this is very unlikely to happen because using a well-known port number as a client port looks suspicious from the outside.

A challenging problem is the possible misclassification of application traffic as port scans. For example, there are applications and services which maintain multiple persistent TCP connections between clients and servers. If the traffic volume is very low, this may result in many flow records containing a small number of packets which does not exceed P_{max}. Examples of such applications which we have found in the analyzed dataset are MS SQL server (TCP port 1433), MySQL server (TCP port 3306), Ident (TCP port 113), Razor database used by Spamassassin (TCP port 2703).

One possibility to filter out this kind of application traffic is to look at the port numbers because the server uses the well-known or registered port of the given service. For this purpose, we already restrict the source ports of the scanning flows to numbers above 1023, hence traffic of services with well-known ports are ignored. Many operation systems choose ephemeral client ports from the range of registered ports between 1024 and 49151. Hence, it is likely that a scanner will use local ports out of this port range as well. Therefore, the only reasonably way to ignore services which are susceptible to the misclassification as port scans is to filter out the corresponding ports individually.

The analysis of the average packet length distribution is very useful to prevent misclassification because varying packet lengths are strong indicators of user or application data being transfered. However, during idle periods in which an application only exchanges empty TCP ACK packets or application-specific heartbeat messages, this criteria fails to distinguish between application traffic and port scans.

During a TCP port scan, each scanning flow may trigger a flow in reverse direction. In practice, however, there is often a firewall which blocks the majority of the scanning flows. Consequently, very few scanning flows reach their destination and provoke reverse traffic. In contrast, established TCP connections result in bidirectional flows unless asymmetric routing is in place. Hence, if the number of scanning flows is much larger than the number of reverse flows, we very likely observe a TCP port scan. As mentioned in Section 10.5.1, this situation applied to the port scans detected in the analyzed flow dataset.

In the case of TCP, it would be very valuable to evaluate the TCP control bits field. This would allow us to recognize flows consisting of TCP SYN packets, which is the type of packet primarily used during port scans.

10.6 Password Guessing

10.6.1 Password Guessing Characteristics

Remote login to specific services often requires a valid user name and password for authentication. If the user enters a certain number of invalid combinations of user name and password, the server usually terminates the connection or session. For example, a typical Secure Shell (SSH) server gives the user three trials before shutting down the TCP connection. In the case of Server Message Block (SMB), the TCP connection is already closed after a single failed login attempt.

With help of automated scripts, hackers are able to test large numbers of different user name and password combinations. These scripts automatically reconnect each time the server has closed the connection and try again with new login data. Often, user names and passwords are taken from a list, which is then called a ‘dictionary attack’. As we do not know if a dictionary is deployed, we use the generic term ‘password guessing’ instead.

```
  Input: TCP flow records of the given time interval, maximum number of
         packets $P_{max} \geq 4$ per flow, minimum number of connections $C_{min}$
  Output: IP addresses of client and server, server port number
1 From all TCP flows with at least 4 and at most $P_{max}$ packets, determine all
  $(srcIp_i, dstIp_i, dstPort_i)$ tuples for which the numbers of flow records
  $\#records_i$ are $\geq C_{min}$;
2 for $i = 1, 2, \ldots$ do
3     Determine the number of TCP flow records $r$ with at least 4 and at
      most $P_{max}$ packets, destination IP address $srcIp_i$, source port $dstPort_i$,
      and source IP address $dstIp_i$;
4     if $r \approx \#records_i$ then
5         Verify if the variation in the numbers of packets per record is small
          in both directions;
6         Report password guessing originating from client $srcIp_i$ to port
          $dstPort_i$ on server $dstIp_i$;
7     endif
8 endfor
```

Alg. 3: Password guessing identification

The number of packets and bytes exchanged per connection is typically small and does not vary much between different connections. According to our observation, failed SSH login attempts result in 10 to 15 packets in each direction, failed SMB login attempts in 6 to 10 packets. Hence, password guessing appears as a large number of similar, short connections between one client and one server. Thereby, the server port remains the same while the port number at the client varies from one connection to the other.

We assume that TCP is used as transport protocol, which is commonly the case for the considered services. As TCP connection establishment and termination require at least three packets in each direction, we can expect that four packets or more are exchanged during password guessing.

10.6.2 Algorithms for Password Guessing Identification

As discussed in the previous section, flows involved in a failed login attempt contain a small number of packets. We restrict our search to flows with at least four packets and at most P_{max} packets. Setting a lower limit for the number of packets enables the distinction between flows which are potentially part of password guessing and flows occurring during a network or port scan because the later ones usually contain only one packet. Regarding the maximum number of packets, $P_{max} = 20$ works fine for our purposes.

Just like network and port scans, the traffic pattern of password guessing can be detected with a few queries to the flow database. Algorithm 3 shows a possible solution. In line 1, we determine all triples of source IP address, destination IP address, and destination port for which the number of TCP flows with at least four and at most P_{max} packets is equal to or larger than C_{min}. C_{min} defines the minimum number of failed login attempts which must be observed in the given time interval in order to consider the traffic as the possible result of password guessing. As a consequence, a small number of failed login attempts, which may occur if a legitimate user has forgotten its password, will not lead to a false alarm.

In SQL, the first query looks as follows:

```
SELECT srcIp, dstIp, dstPort, COUNT(*) AS r
  FROM <flow_table> WHERE proto=6 AND pkts>=4
  AND pkts<=20 GROUP BY srcIp, dstIp, dstPort
  HAVING r>=20;
```

The source IP addresses in the result set belong to potential originators of password guessing. The corresponding destination IP addresses and ports identify the affected services.

The second query (line 3) determines the traffic in reverse direction, again restricting the search to TCP flows with at least four and at most P_{max} packets. The corresponding SQL statement is:

```
SELECT COUNT(*) FROM <flow_table> WHERE proto=6
  AND pkts>=4 AND pkts<=20 AND srcIp=<server_ip>
  AND srcPort=<service_port> AND dstIp=<client_ip>;
```

`<server_ip>`, `<service_port>`, and `<client_ip>` are placeholders for the IP address and port of the service and the IP address of the client initiating the failed login attempts, respectively. If the number of flows is approximately the same in both directions, we can assume that we observe established TCP connections. If the reverse traffic is missing in the dataset, for example due to asymmetric routing, this check will fail as discussed in Section 10.6.3.

Finally, we verify that the variation in the number of packets per flow is small. In fact, we expect very little variation in the case of password guessing as long as the login attempts fail. On the other hand, the numbers of packets are usually more variable if the traffic is not related to login attempts but belongs to a client-server application which uses many small connections. An example is a web browser which establishes multiple HTTP sessions to download objects from a web server. Thus, analyzing the packet count distribution helps distinguishing password guessing from regular application traffic.

10.6.3 Limitations

There are certain situations where password guessing remains unnoticed or where other traffic is falsely regarded as the result of failed login attempts. As we look for a minimum number of connections C_{min} to be observed in the given time interval, password guessing will not be recognized if the login attempts occur at a lower number or rate than expected. The password guessing software may also generate more than P_{max} packets. In order to evade detection, an attacker can enforce this by splitting messages into multiple small TCP segments or by sending additional TCP ACK packets without payload.

Again, asymmetric routing which prevents the observation of both directions of the traffic is a problem. As symmetry in the exchanged traffic is regarded as an essential characteristic, password guessing will not be recognized if only one direction of the connection is monitored. Finally, the traffic differs from the expected pattern if the login is successful. In this case, the TCP connection is not shut down but persists with the exchange of additional data, such as a shell prompt in the case of SSH. Consequently, the number of packets will probably be larger than P_{max}.

It is also possible that a protection mechanism at the server rejects new connection attempts from a client which already failed to provide a valid user name and password combination for a certain number of times. In this case, the resulting flows may contain less than four packets and therefore be ignored by the algorithm. Moreover, password guessing will very likely stop quickly if new TCP connections are rejected. Depending on how quickly the server starts rejecting new connections, the connection count may stay below C_{min} and thus inhibit the password guessing identification.

Traffic of client-server applications using large numbers of short-lived connections risks to be misclassified as password guessing. In particular, this problem occurs with web traffic. In order to accelerate the download of web pages, web browsers open multiple HTTP sessions in parallel to retrieve several objects simultaneously. Furthermore, HTTP sessions are usually closed after the requested objects have been downloaded. Hence, browsing to the next web page results in new TCP connections.

In this case, the packet count distribution helps to distinguish traffic of regular applications and password guessing. Flows of password guessing contain more or less the same number of packets whereas the flow size of application traffic, such as HTTP, is typically much more variable. If this measure is not sufficient, we can filter out traffic of applications resembling password guessing if the corresponding port numbers are known. For example, we can quite reliably exclude HTTP traffic by ignoring flows from and to port 80.

The distinction based on the packet count distribution is less effective in the case of mail delivery with SMTP (Simple Mail Transfer Protocol). In the analyzed dataset, we saw several occurrences of hosts establishing multiple short TCP connections to a mail server (TCP port 25) where the number of packets did not vary considerably. In most of these cases, we observed ten to fifteen packets in each direction. We cannot say if this traffic goes back to legitimate e-mails or spam. The similarity in the flow sizes can be explained by the specific dialog between client and server mandated by the SMTP specification [Kle08]. Filtering out traffic on the well-known SMTP port 25 is the most effective way to prevent misclassification as password guessing.

10.7 Computational and Memory Resources

As we have seen in the previous sections, network and port scans as well as password guessing can be identified with a few queries to the flow database. In this section, we estimate the costs of these queries with respect to computational and memory resources because these two factors directly and indirectly influence the processing time. Therefore, we consider straightforward realizations of the queries only.

The actual resource consumption and processing time of a query depend very much on the performance of the database system and the hardware it is running on. An advanced database system analyzes each query and chooses among different strategies in order to return the result as fast as possible, given the available resources. As an example, it is often possible to reduce the memory consumption of a query at the cost of increasing the computational effort. At the end, this may result in a shorter processing time if the saved memory allows keeping all intermediate results in the system's fast physical memory instead of swapping them to hard disk. A further discussion of such optimizations is beyond the scope of the evaluation.

In all three algorithms, the most demanding query is the first one in line 1. As an example, we consider the first query of the network scan identification algorithm (Algorithm 1). In order to respond to this query, all flows containing at most three packets in the given time interval are aggregated according to distinct triples of protocol, source IP address, and destination port. Hence, the size of the resulting data structure linearly depends on the number of distinct triple values. In addition, the associated destination IP addresses need to be saved separately for each triple. Thus, the number of saved destination IP addresses corresponds to the number of distinct quadruples of protocol, source IP address, destination IP address, and destination port. The maximum memory is required if all n flows matching the given criteria have different combinations of protocol, source IP address, and destination port. This results in n different triples in the first data structure and n destination IP addresses to be saved in total.

For every flow record, the database needs to check if the corresponding triple value and destination IP address already exist in the data structures and eventually insert them as new values. Determining those triple values with at least D_{min} distinct destination IP addresses requires a single run over all triples.

The queries in lines 4 and 10 of Algorithm 1 concern a much smaller subset of the flows because protocol, destination IP address, and source port must match specific values. Therefore, their execution requires less memory and computation than the query in line 1.

In order to reduce the memory consumption of the first query, we could use probabilistic methods to approximate the numbers of distinct destination IP addresses. Probabilistic counting methods have already been discussed in Section 7.3 in the context of cardinality metrics. In the context of network scan identification, their benefit would be lower because we cannot replace the first data structure containing the distinct triple values.

The resources required for the identification of port scans and password guessing according to Algorithms 2 and 3 can be estimated in an analogous way. The results are similar to those of Algorithm 1.

10.8 Discussion

The algorithms presented in this chapter enable the identification of the most frequent relevant anomaly causes found in Chapters 8 and 9. They facilitate the assessment of anomaly detection results and proofed to be very useful for our evaluation and comparison of different residual generation and change detection methods. For the operational deployment of anomaly detection methods, the automated inspection of anomaly causes is even more important than for our research work because network administrators do not have the time to manually get to the bottom of every detected anomaly. Even more, we assume that anomaly detection alone will never be of any practical relevance unless reliable and detailed information about the anomaly causes is provided.

The proposed identification algorithms search flow data for characteristic traffic patterns of relevant incidents. Their deployment is not linked to any specific anomaly detection method. Although we use the algorithms in order to investigate previously detected traffic anomalies, it is also possible to run the identification without any preceding indication of an anomaly. However, continuous searching the collected flow records for known patterns of incidents is costly and requires sufficient resources, especially if the traffic analysis is performed online.

Our implementation of the identification algorithms makes use of a database system in which the flow records are stored. Thanks to the functionality offered by the database system, the algorithms can be reduced to a few SQL queries. Tools using file-based flow repositories, such as *SiLK*, *flow-tools*, and *nfdump*, may be faster under certain conditions, yet lack flexibility regarding the supported types of queries. On the other hand, many database systems implement different optimization strategies and cache results of previous queries, which may significantly speed up consecutive queries to the same set of flow records. A performance evaluation of the flow database approach is an interesting subject of future work.

The proposed set of algorithms can be extended in order to identify more types of incidents. As an example, an algorithm could be developed to recognize hosts which seem to emit a lot of spam e-mails. Schatzmann et al. studied the fingerprint of SMTP traffic in the measured flow data [SBS09] and found flow properties which enable the discrimination of rejected and accepted mail deliveries. Based on these results, it should be possible to identify potential spammers as hosts for which more e-mails are rejected than accepted due to anti-spam mechanisms installed on the mail servers.

Part IV

CONCLUSION

11. CONCLUSIONS AND FUTURE DIRECTIONS

11.1 Summary of Results

In the introduction of this dissertation, we justified the necessity of traffic measurements and analysis by the openness and heterogeneity of the Internet. Network operators have very little control on the utilization of their network and do not know exactly which kinds of applications and services generate the transported traffic. In addition, it is very difficult to prevent harmful or unwanted traffic from entering the network. Therefore, traffic measurements have become an essential source of information. In the backbone networks of Internet service providers (ISPs), such measurements are commonly performed at the level of flows.

This dissertation dealt with the analysis of the resulting flow-level measurement data in order to detect traffic anomalies and inspect their causes. Traffic anomalies are often equated with harmful traffic, such as attacks and worm propagation. However, there are many other causes of traffic anomalies, and not all of them are of interest for the network administrator. Therefore, our goal was to find anomaly detection methods which deliver a large proportion of relevant alarms.

Time series of multiple traffic metrics represented the basis of our traffic analysis. Time series scale in the presence of large amounts of traffic because the processing complexity per time interval is independent of traffic volume and composition. For the purpose of anomaly detection, we adopted and combined various statistical methods of statistical process control, time-series analysis, and principal component analysis (PCA). The evaluation of the different methods was based on flow data collected in an ISP network. Therefore, the results can be considered as representative for real networks.

For anomaly detection in the time series of a single metric, **exponential smoothing** combined with the **Shewhart control chart of individuals** turned out to be the most effective combination of the considered statistical methods. The relevance of the detected alarms, however, depended very much on the selected traffic metrics and the analyzed part of traffic. Most of the alarms detected in the number of distinct source and destination IP addresses and ports were classified as relevant whereas anomalous changes in the numbers of bytes and packets were often caused by irregular but uninteresting data transfers. Very good detection results could be achieved by focusing on selected parts of traffic which were less affected by irrelevant anomalies.

Apart from single-metric approaches, we evaluated different multi-metric anomaly detection methods based on PCA. Best detection results were achieved with the T^2 **and** T_H^2 **control charts** using **batch-mode PCA** and **robust M-estimators**. This method is more complex than single-metric anomaly detection and has the disadvantage that training data is required to estimate the correlation between different metrics. On the other hand, all traffic metrics can be monitored with a single control chart, achieving a large proportion of relevant alarms. The application of incremental PCA was an attempt to jointly consider correlation among different metrics and temporal dependencies in the residual generation process. However, this approach turned out to be very complex and did not produce convincing detection results.

Our evaluation showed that forecasting techniques as well as PCA allow transforming the original measurement time series into residuals or statistics which are much less affected by serial correlation and systematic changes. However, none of these modeling approaches is able to explain the entire variation of normal network traffic. As a consequence, the variables monitored in the control charts must not be assumed to be completely free of serial correlation and non-stationarities in the in-control state. Thus, control limits cannot be calculated for a predefined false positive rate but need to be determined empirically by the network administrator.

In order to facilitate the identification of anomaly causes, we developed three algorithms which search the original flow records for patterns of network scans, port scans, and password guessing. These three types of incidents represented the most frequent causes of relevant anomalies in the analyzed dataset. Using sophisticated patterns reflecting multiple traffic properties, we were able to significantly reduce the risk of false identifications, that is traffic being wrongly identified as a scan or password guessing.

An important property of the examined anomaly detection methods is that they can be easily deployed in an operational network for online analysis of flow data. One possibility is to integrate them into the *TOPAS* framework [MC07] which has been explicitly conceived for real-time traffic analysis and attack detection. As part of this framework, the methods evaluated in this dissertation can be combined with other detection techniques, such as signature detection in packet-level measurement data (see Section 2.4.2).

11.2 Evaluation of Achievements

The research work conducted in the scope of this dissertation provided the evaluation and comparison of different traffic anomaly detection techniques. The evaluation was based on a thorough understanding of the fundamentals and prerequisites of the statistical methods. Instead of simulated or generated traffic, we used traffic measurement data collected in the network of an

ISP in order to obtain realistic results.

An important aspect of the evaluation was to assess the applied methods as well as the detected anomalies from the **point of view of the network administrator**. Thereby, we started from the assumption that network administrators do not have the time and expertise to tune the parameters of complex detection methods. Also, they do not want to be bothered with irrelevant alarms which go back to significant but harmless traffic variations. Finally, the notification of an anomaly should be enriched with additional information about the responsible flows and the involved hosts. In the ideal case, the most likely cause of an anomaly would be directly reported to relieve the administrator of time-consuming manual inquiries.

Based on these considerations, we defined four criteria in Section 1.2 according to which the studied traffic anomaly detection approaches should be evaluated. In the following, we judge the outcome of the dissertation and show that the proposed combination of residual generation techniques, change detection, and anomaly identification algorithms fulfills the given criteria in a satisfactory manner.

Ease of deployment

Exponential smoothing and the Shewhart control chart of individuals can be implemented very easily. The Shewhart control chart of individuals can be realized with constant or adaptive control limits. Using constant control limits, there exist only two configuration parameters: the smoothing constant of exponential smoothing α and the level of the control limits L. In the case of adaptive control limits, a second smoothing constant ρ controls the updates of the EWMS estimator. $\alpha = 0.5$ and $\rho = 0.01$ turned out to be appropriate values under various conditions, thus there is no need to change these settings. On the other hand, the level of the control limits directly influences the number of triggered alarms and therefore represents the primary knob for adjustments.

More demanding is the implementation and deployment of multi-metric anomaly detection based on batch-mode PCA. Calculating the PCs requires the estimation of the covariance matrix and the subsequent computation of eigenvalues and eigenvectors. Therefore, training data is needed which must not contain any long-lasting anomalies. Sporadic outliers can be tolerated if the robust M-estimator is used to estimate mean and covariances. The parameterization, on the other hand, is relatively simple. Again, the main knob to control the number of alarms is the level of the control limit of the T^2 or T^2_H control chart. In the case of T^2_H, we need to decide on the number of principal components (PCs) retained in the normal subspace. According to our experience, however, the choice of this parameter does not have a big influence on the detection results.

Justification of complex methods

As a second criterion, we postulated that the application of complex methods should be justified by significant improvements regarding the detection results compared to simpler methods. Exponential smoothing and Shewhart control chart are the least complex univariate statistical methods considered. More sophisticated methods, such as residual generation using Holt-Winters forecasting, did not lead to improvements but required the configuration of additional parameters.

Less clear is the decision between single-metric and multi-metric anomaly detection. Certain types of relevant anomalies, such as network scans, can be very reliably detected with single-metric methods at a low level of irrelevant alarms. On the other hand, the T^2 and T_H^2 statistics enable the detection of a wide spectrum of relevant anomalies with a single control chart. Thus, the user has to decide whether the advantages of the T^2 and T_H^2 control charts justify the additional complexity as well as the necessary provision of appropriate training data.

Relevance of the detected anomalies

In the case of single-metric anomaly detection, the relevance of the detected anomalies depends on the considered traffic metric and the analyzed part of traffic. Regarding the overall traffic, for example, most relevant anomalies were found in the number of distinct IP addresses or ports. On the other hand, the analysis of ICMP and SMB traffic resulted in a large proportion of relevant alarms in the other traffic metrics as well. The multi-metric T^2 and T_H^2 control charts triggered mostly relevant alarms over a wide range of control limits, regardless of the analyzed part of traffic.

Since some types of relevant anomalies can be easier detected by the single-metric and others by the multi-metric approach, it can make sense to simultaneously apply single-metric and multi-metric anomaly detection to detect more relevant anomalies. In this case, the proportion of distinct relevant alarms may increase or decrease, depending on the overlap between the sets of relevant and irrelevant anomalies detected by both approaches.

Identification of anomaly causes

In general, identifying anomaly causes is beyond the scope of anomaly detection, which is a major disadvantage compared to the misuse detection approach which explicitly looks for known types of incidents. In the case of single-metric anomaly detection, knowledge about the metrics in which an anomaly has been found allows us to at least narrow the set of potential causes down, as discussed in Section 8.5. For example, an anomalously large number of distinct destination IP addresses is a sign of a potential network scan. Apart from such vague indicators, however, the considered anomaly detection methods may not offer any further information.

In order to compensate this severe shortcoming, we presented various algorithms for the automated identification of common causes of anomalies. Each algorithm analyzes the original flow records collected in the time interval of the detected anomaly and looks for characteristic traffic patterns of a known type of incident. In this dissertation, we focused on the identification of network scans, port scans, and password guessing, yet the approach can be extended to other kinds of incidents as well.

Obviously, the usage of identification algorithms is limited to previously known types of incidents which can be mapped to specific traffic patterns. However, as most relevant anomalies go back to a small number of common causes, most of the interesting alarms can be identified automatically.

11.3 Future Directions

Traffic anomaly detection has been a research topic for many years. Although a lot of different methods have been developed and evaluated in the academic world, traffic anomaly detection still plays a subordinate role for the monitoring and management of operational networks. This discrepancy suggests that most of the proposed solutions somehow do not meet the actual requirements of network administrators.

In our opinion, there are several problems which inhibit the practical deployment of traffic anomaly detection. First of all, many anomaly detection methods are quite complex and difficult to parameterize. Secondly, many anomaly detection methods tend to trigger a large number of alarms which are of no interest for the network administrator. Such irrelevant anomalies are often related to the unpredictable behavior of users and applications. Thirdly, the network administrator is mainly interested in the causes of an anomaly and not so much in its statistical effect on traffic properties. Hence, the pure knowledge of anomalous traffic observed at a certain point in time is quite worthless in practice.

In this dissertation, we addressed these three issues. We showed that good detection results can be achieved with methods which are very easy to deploy. We evaluated and compared the examined methods with respect to the number and proportion of relevant anomalies detected in the analyzed dataset. Finally, we developed three pattern-based identification algorithms to examine if an anomaly can be associated to specific kinds of causes. Starting from our experiences and results, we see two promising directions of future research, as sketched in the following.

The first direction concerns the **input data of traffic anomaly detection**. Our evaluation results show that the relevance of the detected anomalies strongly depends on the considered traffic metrics and the analyzed parts of traffic. This confirms our initial assumption, mentioned in

Section 1.2, that more emphasis should be placed on the choice of appropriate input data instead of trying out ever new statistical methods.

One approach to find better input data is to look for appropriate traffic metrics. For example, we showed that anomalies found in cardinality metrics (i.e., the numbers of distinct IP addresses or ports) have a high probability of being caused by relevant incidents. In the past, researchers calculated entropy values of these flow keys instead of cardinalities [LCD05, TBSM09]. Hence, it would be interesting to compare the detection results of cardinality metrics and entropy metrics in order to see if the more complex calculation of entropy values is beneficial. Furthermore, it would be worth investigating if better metrics can be derived from the measurement data, for example by using ratios of different cardinality values.

Another approach, which we expect to improve the detection results even more, is to apply anomaly detection methods to specific parts of traffic which are barely affected by irrelevant anomalies. For this purpose, it is very helpful to have a rough notation about possible relevant incidents and how they affect the analyzed measurement data. Based on such considerations, we separated the ICMP and SMB traffic of our dataset and showed that the large majority of anomalies found in these parts of traffic are relevant. Very probably, there exist other protocols and applications for which a separate analysis of the corresponding traffic would be promising. Alternatively, it can make sense to analyze the overall traffic from which the traffic of specific applications has been removed. This can be useful if a legitimate application causes a lot of irrelevant alarms, such as web (HTTP) or DNS. In either case, the selected or removed parts of traffic need to be identifiable by packet header fields or flow keys, such as the transport protocol and port numbers, to enable an easy filtering of the packets or flows.

The second direction of future research concerns the automated **pattern-based identification of incidents**. We have presented identification algorithms for network scans, port scans, and password guessing. Additional algorithms can be developed for other incidents as well. For example, it should be possible to identity hosts trying to emit a lot of spam e-mails, especially if many of these e-mails are rejected by mail servers, which results in many short connections.

The usage of a flow database simplifies the realization of identification algorithms because complex queries can be easily expressed in a high-level query language, such as SQL. On the other hand, the deployment of general purpose database systems likely requires more computational and memory resources than specialized programs using file-based flow repositories. Therefore, we need to evaluate the performance of flow databases with respect to the processing time of the queries as well as the maximum rate at which new records can be written into the database. The insertion speed is important because flow records have to be written into the database as fast as they arrive at the collector. Otherwise, we cannot use the flow database for real-time traffic analysis. In fact, the dataset analyzed in this dissertation does not cause any performance problems. In other networks, however, the number of flows can be much higher.

If the causes of most relevant anomalies can be identified by specific traffic patterns, the question arises whether these patterns enable the detection of such incidents without using any anomaly detection method. To do so, the collected flow records need to be continuously searched for known patterns in real-time. If this was possible, we could build a purely pattern-based detection system for known types of incidents. Obviously, unknown types of incidents would remain unnoticed because such a system follows the misuse detection approach. However, as the utility of anomaly detection for the network administrator is limited anyway, the pattern-based analysis of flow data may turn out to be the better solution for practical deployment.

ACKNOWLEDGMENT

This dissertation is the result of several years of studying and working on traffic measurement and anomaly detection techniques. Discussions with other researchers allowed me to get a deeper insight and better understanding of this topic. Another immense source of knowledge and experience was the collaboration with numerous students during their student research projects and diploma theses. I express my sincere gratitude to all of these persons.

My special thanks are due to Professor Dr. Georg Carle for supervising and supporting my dissertation project, to Professor Anja Feldmann, Ph.D., and Professor Dr. Andreas Herkersdorf for providing valuable comments and suggestions, and to Professor Dr. Baumgarten for kindly organizing the examination procedure. Further thanks go to my colleagues for their very helpful feedback and advice during the rehearsals of my final presentation.

BIBLIOGRAPHY

[AKM$^+$05] William Aiello, Charles Kalmanek, Patrick McDaniel, Subhabrata Sen, Oliver Spatscheck, and Jacobus Van der Merwe. Analysis of communities of interest in data networks. In *Proc. of Passive and Active Measurement Workshop (PAM) 2005*, Boston, MA, USA, March 2005.

[APT07] Mark Allman, Vern Paxson, and Jeff Terrell. A brief history of scanning. In *Proc. of Internet Measurement Conference (IMC) 2007*, San Diego, CA, USA, October 2007.

[BB83] Michèle Basseville and Albert Benveniste. Sequential detection of abrupt changes in spectral characteristics of digital signals. *IEEE Transactions on Information Theory*, 29(5):709–724, September 1983.

[BD93] Boris E. Brodsky and Boris S. Darkhovsky. *Nonparametric Methods in Change-Point Problems*, volume 243 of *Mathematics and its applications.* Kluwer Academic Publishers, 1993.

[BKPR02] Paul Barford, Jeffery Kline, David Plonka, and Amos Ron. A signal analysis of network traffic anomalies. In *Proc. of ACM SIGCOMM Internet Measurement Workshop 2002*, Pittsburgh, PA, USA, November 2002.

[BKRT01] Rudolf B. Blazek, Hongjoong Kim, Boris L. Rozovskii, and Alexander G. Tartakovsky. A novel approach to detection of denial-of-service attacks via adaptive sequential and batch-sequential change-point detection methods. In *Proc. of IEEE Workshop on Information Assurance and Security*, West Point, NY, USA, June 2001.

[BM06] Lothar Braun and Gerhard Münz. Netzbasierte Angriffs- und Anomalieerkennung mit TOPAS. In Ulrich Flegel, editor, *GI FG SIDAR Graduierten-Workshop über Reaktive Sicherheit (SPRING), SIDAR-Report SR-2006-01*, Berlin, Germany, July 2006.

[BMC10] Lothar Braun, Gerhard Münz, and Georg Carle. Packet sampling for worm and botnet detection in TCP connections. In *Proc. of IEEE/IFIP Network Operations and Management Symposium (NOMS) 2010*, Osaka, Japan, April 2010.

[BMG05] James R. Binkley, John McHugh, and Carrie Gates. Locality, network control and anomaly detection. Technical Report 04-04, Portland State University, Computer Science Department, January 2005.

[BMP07] Daniela Brauckhoff, Martin May, and Bernhard Plattner. Comparison of anomaly signal quality in common detection metrics. In *Proc. of ACM SIGMETRICS MineNet Workshop*, San Diego, CA, USA, June 2007.

[BN93] Michèle Basseville and Igor V. Nikiforov. *Detection of abrupt changes: Theory and application.* Prentice Hall, 1993.

[Bru00] Jake D. Brutlag. Aberrant behavior detection in time series for network monitoring. In USENIX Association, editor, *Proc. of USENIX Conference on System Administration (LISA) 2000*, New Orleans, LA, USA, December 2000.

[BSM09] Daniela Brauckhoff, Kave Salamatian, and Martin May. Applying PCA for traffic anomaly detection: Problems and solutions. In *Proc. of IEEE Conference on Computer Communications (INFOCOM) 2009*, Rio de Janeiro, Brazil, April 2009.

[BTS06] Laurent Bernaille, Renata Teixeira, and Kavé Salamatian. Early application identification. In *Proc. of ACM International Conference On Emerging Networking Experiments And Technologies (CoNEXT) 2006*, Lisboa, Portugal, December 2006.

[CAJM10] Benoit Claise, Paul Aitken, Andrew Johnson, and Gerhard Münz. IPFIX Export per SCTP Stream. Internet-Draft (work in progress), draft-ietf-ipfix-export-per-sctp-stream-07, March 2010.

[Cat66] Raymond Bernard Cattell. The scree test for the number of factors. *Multivariate Behavioral Research*, 1(2):245–276, 1966.

[CB97] Mark E. Crovella and Azer Bestavros. Self-similarity in World Wide Web traffic: Evidence and possible causes. *IEEE/ACM Transactions on Networking*, 5(6):835–846, December 1997.

[CBS+08] Benoit Claise, Stewart Bryant, Ganesh Sadasivan, Simon Leinen, Thomas Dietz, and Brian H. Trammell. Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of IP Traffic Flow Information. RFC 5101 (Proposed Standard), January 2008.

[Cha03] Cristopher Chatfield. *The analysis of time series: an introduction.* CRC Press LLC, 6th edition, 2003.

[Cis10a] Cisco Flexible NetFlow Homepage. http://www.cisco.com/go/fnf, 2010.

[Cis10b] Cisco NetFlow Homepage. http://www.cisco.com/go/netflow, 2010.

[CJQ09] Benoit Claise, Andrew Johnson, and Jürgen Quittek. Packet Sampling (PSAMP) Protocol Specifications. RFC 5476 (Proposed Standard), March 2009.

[CMZ06] Graham Cormode, S. Muthu Muthukrishnan, and Wei Zhuang. What's different: Distributed, continuous monitoring of duplicate-resilient aggregates on data streams. In *Proc. of IEEE International Conference on Data Engineering*, pages 20–31, Atlanta, GA, USA, April 2006.

[CPA09] Vasilis Chatzigiannakis, Symeon Papavassiliou, and Georgios Androulidakis. Improving network anomaly detection effectiveness via an integrated multi-metric-multi-link (M3L) PCA-based approach. *Security and Communication Networks*, 2(3):289–304, May 2009.

[CPAM06] Vasilis Chatzigiannakis, Symeon Papavassiliou, Georgios Androulidakis, and Basil S. Maglaris. On the realization of a generalized data fusion and network anomaly detection framework. In *Proc. of Communication Systems, Networks and Digital Signal Processing (CSNDSP) 2006*, Patras, Greece, July 2006.

[CR96] Nuno Crato and Bonnie K. Ray. Model selection and forecasting for long-range dependent processes. *Journal of Forecasting*, 15(2):107–125, March 1996.

[CRM00] João B. D. Cabrera, B. Ravichandran, and Raman K. Mehra. Statistical traffic modeling for network intrusion detection. In *Proc. of International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS) 2000*, San Francisco, CA, USA, August 2000.

[Cro88] Ronald B. Crosier. Multivariate generalizations of cumulative sum quality-control schemes. *Technometrics*, 30(3):291–303, 1988.

[CSVD04] Benoit Claise, Ganesh Sadasivan, Vamsi Valluri, and Martin Djernaes. Cisco Systems NetFlow Services Export Version 9. RFC 3954 (Informational), October 2004.

[Dac08] Marc Dacier. Leurré.com: a worldwide distributed honeynet, lessons learned after 4 years of existence. In *Presentation at Terena Networking Conference*, Bruges, Belgium, May 2008.

[DAR01] DARPA Intrusion Detection Evaluation 1998-2000 Homepage. http://www.ll.mit.edu/IST/ideval/index.html, 2001.

[DC05] Falko Dressler and Georg Carle. HISTORY - High Speed Network Monitoring and Analysis. In *Proc. of IEEE Conference on Computer Communications (INFOCOM) 2005, Poster Session*, March 2005.

[DCA$^+$09] Thomas Dietz, Benoit Claise, Paul Aitken, Falko Dressler, and Georg Carle. Information Model for Packet Sampling Exports. RFC 5477 (Proposed Standard), March 2009.

[DCC$^+$09] Nevil Duffield, Derek Chiou, Benoit Claise, Albert Greenberg, Matthias Grossglauser, and Jennifer Rexford. A Framework for Packet Selection and Reporting. RFC 5474 (Informational), March 2009.

[DF03] Marianne Durand and Philippe Flajolet. Loglog counting of large cardinalities. In *Proc. of Annual European Symposium on Algorithms (ESA) 2003*, pages 605–617, Budapest, Hungary, September 2003.

[DG01] Nick Duffield and Markus Grossglauser. Trajectory sampling for direct traffic observation. *IEEE/ACM Transactions on Networking*, 9(3):280–292, June 2001.

[DHKR09] Nick Duffield, Patrick Haffner, Balachander Krishnamurthy, and Haakon Ringberg. Rule-based anomaly detection on IP flows. In *Proc. of IEEE Conference on Computer Communications (INFOCOM) 2009*, Rio de Janeiro, Brazil, April 2009.

[DIA06] DIADEM Firewall Homepage. http://www.diadem-firewall.org, 2006.

[DKCM10] Thomas Dietz, Atsushi Kobayashi, Benoit Claise, and Gerhard Münz. Definitions of Managed Objects for IP Flow Information Export. RFC 5815 (Proposed Standard), April 2010.

[DLT04] Nick Duffield, Carsten Lund, and Mikkel Thorup. Flow sampling under hard resource constraints. In *Proc. of ACM Joint International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS-Performance) 2004*, New York, NY, USA, June 2004.

[DM06] Falko Dressler and Gerhard Münz. Flexible flow aggregation for adaptive network monitoring. In *Proc. of IEEE LCN Workshop on Network Measurements 2006*, Tampa, FL, USA, November 2006.

[DMBC09] Hui Dai, Gerhard Münz, Lothar Braun, and Georg Carle. TCP-Verkehrsklassifizierung mit Markov-Modellen. In *Proc. of GI/ITG-Workshop Leistungs-, Zuverlässigkeits- und Verlässlichkeitsbewertung von Kommunikationsnetzen und Verteilten Systemen (MMBnet) 2009*, Hamburg, Germany, September 2009.

[DP05] Thomas Dübendorfer and Bernhard Plattner. Host behaviour based early detection of worm outbreaks in internet backbones. In *Proc. of IEEE International Workshops on Enabling Technologies*, Linköping, Sweden, June 2005.

[DS01] Morris H. DeGroot and Mark J. Schervish. *Probability and Statistics*. Addison Wesley, 3rd edition, 2001.

[Duf04] Nick Duffield. Sampling for passive internet measurement: A review. *Statistical Science*, 10(3):472–498, 2004.

[EBC$^+$06] Rob Enns, Andy Bierman, Ken Crozier, Ted Goddard, Eliot Lear, Phil Shafer, Steve Waldbusser, and Margaret Wasserman. NETCONF Configuration Protocol. RFC 4741 (Standards Track), December 2006.

[EBR03] James P. Early, Carla E. Brodley, and Catherine Rosenberg. Behavioral authentication of server flows. In *Proc. of Applied Computer Security Associates (ACSA) 2003*, Las Vegas, NV, USA, December 2003.

[ENV07] Majid Eyvazian, S. G. Jalali Naini, and A. Vaghefi. Monitoring process variability using exponentially weighted moving sample variance control charts. *The International Journal of Advanced Manufacturing Technology (Online First)*, 39(3–4):261–270, October 2007.

[ETGTDV04a] Juan M. Estevez-Tapiador, Pedro Garcia-Teodoro, and Jesus E. Diaz-Verdejo. Anomaly detection methods in wired networks: a survey and taxonomy. *Computer Communications*, 27(16):1569–1584, October 2004.

[ETGTDV04b] Juan M. Estevez-Tapiador, Pedro Garcia-Teodoro, and Jesus E. Diaz-Verdejo. Measuring normality in HTTP traffic for anomaly-based intrusion detection. *Computer Networks*, 45(2):175–193, June 2004.

[EVF03] Cristian Estan, George Varghese, and Mike Fisk. Bitmap algorithms for counting active flows on high speed links. In *Proc. of Internet Measurement Conference (IMC) 2003*, Miami Beach, FL, USA, October 2003.

[FFGM07] Philippe Flajolet, Éric Fusy, Olivier Gandouet, and Frédéric Meunier. HyperLoglog: the analysis of a near-optimal cardinality estimation algorithm. In *Proc of International Conference on the Analysis of Algorithms (AofA) 2007*, Juan-les-pins, France, June 2007.

[FGW98] Anja Feldmann, Anna C. Gilbert, and Walter Willinger. Data networks as cascades: investigating the multifractal nature of Internet WAN traffic. In *Proc. of ACM Conference of the Special Interest Group on Data Communication (SIGCOMM) 1998*, pages 42–55, Vancouver, Canada, September 1998.

[FGWK98] Anja Feldmann, Anna C. Gilbert, Walter Willinger, and T.G. Kurtz. The changing nature of network traffic: scaling phenomena. *ACM SIGCOMM Computer Communication Review*, 28(2):5–29, April 1998.

[FM85] Philippe Flajolet and G. Nigel Martin. Probabilistic counting algorithms for data base applications. *Journal of Computer and System Sciences*, 31(2):182–209, 1985.

[FR00] Mark Fullmer and Steve Romig. The OSU flow-tools package and Cisco NetFlow logs. In *Proc. of USENIX Conference on System Administration (LISA) 2000*, pages 291–304, New Orleans, LA, USA, December 2000.

[FSBK03] Laura Feinstein, Dan Schnackenberg, Ravindra Balupari, and Darrell Kindred. Statistical approaches to DDoS attack detection and response. In *Proc. of DARPA Information Survivability Conference and Exposition (DISCEX) 2003*, pages 303–315, Washington, DC, USA, 2003.

[GB05] Carrie Gates and Damon Becknel. Host anomalies from network data. In *Proc. of IEEE Systems, Man and Cybernetics Information Assurance Workshop*, West Point, NY, USA, June 2005.

[GCD$^+$04] Carrie Gates, Michael Collins, Michael Duggan, Andrew Kompanek, and Mark Thomas. More netflow tools: For performance and security. In *Proc. of Large Installation System Administration Conference (LISA) 2004*, Atlanta, GA, USA, November 2004.

[GJ06] Everette S. Gardner Jr. Exponential smoothing: The state of the art–part ii. *International Journal of Forecasting*, 22(4):637–666, October 2006.

[GL96] Gene H. Golub and Charles F. Van Loan. *Matrix Computations.* Johns Hopkins University Press, 3rd edition, 1996.

[GNU10] GNU Octave Homepage. http://www.octave.org, 2010.

[GR52] Meyer A. Girshick and Herman Rubin. A bayes approach to a quality control model. *Annals of Mathematical Statistics*, 23(1):114–125, 1952.

[GT06] Carrie Gates and Carol Taylor. Challenging the anomaly detection paradigm: a provocative discussion. In *Proc. of ACM Workshop on New Security Paradigms 2006*, Schloss Dagstuhl, Germany, September 2006.

[Haj05] Hassan Hajji. Statistical analysis of network traffic for adaptive faults detection. *IEEE Transactions on Neural Networks*, 16(5), September 2005.

[Haw74] Douglas M. Hawkins. The detection of errors in multivariate data using principal components. *Journal of the American Statistical Association*, 69(346), June 1974.

[HFW01] Polly Huang, Anja Feldmann, and Walter Willinger. A non-intrusive, wavelet-based approach to detecting network performance problems. In *Proc. of ACM SIGCOMM Internet Measurement Workshop 2001*, San Francisco, CA, USA, November 2001.

[Hig10] High-Availability.Com (HAC) Homepage. http://www.high-availability.com, 2010.

[HJ97] Cynthia S. Hood and Chuanyi Ji. Proactive network fault detection. In *Proc. of IEEE Conference on Computer Communications (INFOCOM) 1997*, pages 147–1155, Kobe, Japan, April 1997.

[HM93] Donald S. Holmes and A. Erhan Mergen. Improving the performance of T-square control chart. *Quality Engineering*, 5(4):619–625, 1993.

[HMM98] Peter M. Hall, David Marshall, and Ralph R. Martin. Incremental eigenanalysis for classification. In *Proc. of British Machine Vision Conference*, pages 286–295, Southampton, UK, 1998.

[Hot31] Harold Hotelling. A generalization of student's ratio. *Annals of Mathematical Statistics*, (2):360–378, 1931.

[HZS98] Joseph L. Hellerstein, Fan Zhang, and Perwez Shahabuddin. Characterizing normal operation of a web server: Application to workload forecasting and capacity planning. In *Proc. of International Computer Measurement Group (CMG) Conference*, Anaheim, CA, USA, December 1998.

[IG99] Makoto Iguchi and Shigeki Goto. Detecting malicious activities through port profiling. *IEICE Transactions on Information and Systems*, E82-D(4), April 1999.

[Ise97] David S. Isenberg. The rise of the stupid network. *Computer Telephony*, pages 16–26, April 1997.

[Jac91] J. Edward Jackson. *A User's Guide to Principal Components*. Wiley-Interscience, 1991.

[Jol02] Ian T. Jolliffe. *Principal Component Analysis*. Springer-Verlag New York, Inc., 2nd edition, 2002.

[JPBB04] Jaeyeon Jung, Vern Paxson, Arthur W. Berger, and Hari Balakrishnan. Fast portscan detection using sequential hypothesis testing. In *Proc. of IEEE Symposium on Security and Privacy*, Oakland, CA, USA, May 2004.

[JR86] Raj Jain and Shawn A. Routhier. Packet trains–measurements and a new model for computer network traffic. *IEEE Journal on Selected Areas in Communications*, 4(6):986–995, September 1986.

[Jun10] Juniper Homepage. http://www.juniper.net, 2010.

[KCF$^+$08] Hyun-chul Kim, Kimberly Claffy, Marina Fomenkov, Dhiman Barman, Michalis Faloutsos, and KiYoung Lee. Internet traffic classification demystified: Myths, caveats, and the best practices. In *Proc. of ACM International Conference On Emerging Networking Experiments And Technologies (CoNEXT) 2008*, Madrid, Spain, December 2008.

[KDD99] KDD Cup 1999 Homepage. http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html, 1999.

[KGE06] Andreas Kind, Dieter Gantenbein, and Hiroaki Etoh. Relationship discovery with netflow to enable business-driven IT management. In *Proc. of IEEE/IFIP International Workshop on Business-Driven IT Management (BDIM) 2006*, Vancouver, Canada, April 2006.

[Kle08] John Klensin. Simple Mail Transfer Protocol. RFC 5321 (Draft Standard), October 2008.

[KME05] Ken Keys, David Moore, and Cristian Estan. A robust system for accurate real-time summaries of internet traffic. In *Proc. of International Conference on Measurement & Modeling of Computer Systems (SIGMETRICS) 2005*, Banff, Canada, June 2005.

[KPF05] Thomas Karagiannis, Konstantina Papagiannaki, and Michalis Faloutsos. BLINC: Multilevel traffic classification in the dark. In *Proc. of ACM Conference of the Special Interest Group on Data Communication (SIGCOMM) 2005*, Philadelphia, PA, USA, August 2005.

[KR06] Seong Soo Kim and A. L. Narasimha Reddy. An evaluation of the effectiveness of measurement based anomaly detection techniques. In *Proc. of Workshop on Assurance in Distributed Systems and Networks (ADSN) 2006*, Lisbon, Portugal, July 2006.

[KRT04] Hongjoong Kim, Boris L. Rozovskii, and Alexander G. Tartakovsky. A nonparametric multichart CUSUM test for rapid detection of DoS attacks in computer networks. *International Journal of Computing & Information Sciences*, 2(3):149–158, December 2004.

[KTK02] Christopher Krügel, Thomas Toth, and Engin Kirda. Service specific anomaly detection for network intrusion detection. In *Proc. of Symposium on Applied Computing (SAC) 2002*, Madrid, Spain, March 2002.

[LBC^{+}06] Xin Li, Fang Bian, Mark Crovella, Christophe Diot, Ramesh Govindan, Gianluca Iannaccone, and Anukool Lakhina. Detection and identification of network anomalies using sketch subspaces. In *Proc. of Internet Measurement Conference (IMC) 2006*, Rio de Janeiro, Brazil, October 2006.

[LC02] Xiaoming Liu and Tsuhan Chen. Shot boundary detection using temporal statistics modeling. In *Proc. of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP) 2002*, Orlando, FL, USA, May 2002.

[LCD04a] Anukool Lakhina, Mark Crovella, and Christiphe Diot. Characterization of network-wide anomalies in traffic flows. In *Proc. of Internet Measurement Conference (IMC) 2004*, pages 201–206, Taormina, Italy, October 2004.

[LCD04b] Anukool Lakhina, Mark Crovella, and Christophe Diot. Diagnosing network-wide traffic anomalies. In *Proc. of ACM Conference of the Special Interest Group on Data Communication (SIGCOMM) 2004*, Portland, OR, USA, August 2004.

[LCD05] Anukool Lakhina, Mark Crovella, and Christophe Diot. Mining anomalies using traffic feature distributions. In *Proc. of ACM Conference of the Special Interest Group on Data Communication (SIGCOMM) 2005*, Philadelphia, PA, USA, August 2005.

[Lei04] Simon Leinen. Evaluation of Candidate Protocols for IP Flow Information Export (IPFIX). RFC 3955 (Informational), October 2004.

[Li04] Yongmin Li. On incremental and robust subspace learning. *Pattern Recognition*, 37(7):1509–1518, July 2004.

[Lor71] Gary Lorden. Procedures for reacting to a change in distribution. *Annals of Mathematical Statistics*, 42(6):1897–1908, 1971.

[LPC$^+$04] Anukool Lakhina, Konstantina Papagiannaki, Mark Crovella, Christiphe Diot, Eric D. Kolaczyk, and Nina Taft. Structural analysis of network traffic flows. In *Proc. of ACM Joint International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS-Performance) 2004*, New York, NY, USA, June 2004.

[LR02] Christopher Leckie and Kotagiri Ramamohanarao. A Probabilistic Approach to Detecting Network Scans. In *Proc. of IEEE/IFIP Network Operations and Management Symposium (NOMS) 2002*, Florence, Italy, April 2002.

[LSMD06] Ronny T. Lampert, Christoph Sommer, Gerhard Münz, and Falko Dressler. Vermont - a versatile monitoring toolkit for IPFIX and PSAMP. In *Proc. of IEEE/IST Workshop on Monitoring, Attack Detection and Mitigation (MonAM) 2006*, Tübingen, Germany, September 2006.

[LTWW93] Will E. Leland, Murad S. Taqqu, Walter Willinger, and Daniel V. Wilson. On the self-similar nature of Ethernet traffic. In *Proc. of ACM Conference of the Special Interest Group on Data Communication (SIGCOMM) 1993*, San Francisco, CA, USA, September 1993.

[LWCR92] Cynthia A. Lowry, William H. Woodall, Charles W. Champ, and Steven E. Rigdon. A multivariate exponentially weighted moving average control chart. *Technometrics*, 34(1):46–53, February 1992.

[LYVCQ00] Weihua Li, H. Henry Yue, Sergio Valle-Cervantes, and S. Joe Qin. Recursive PCA for adaptive process monitoring. *Journal of Process Control*, 10(5):471–486, October 2000.

[MADC06] Gerhard Münz, Albert Antony, Falko Dressler, and Georg Carle. Using Netconf for configuring monitoring probes. In *Proc. of IEEE/IFIP Network Operations & Management Symposium (NOMS) 2006, Poster Session*, Vancouver, Canada, April 2006.

[Mal05] Bjarte Malmedal. Using netflows for slow portscan detection. Master's thesis, Gjovik University College, 2005.

[MC07] Gerhard Münz and Georg Carle. Real-time analysis of flow data for network attack detection. In *Proc. of IFIP/IEEE Symposium on Integrated Management (IM) 2007*, Munich, Germany, May 2007.

[MC08a] Gerhard Münz and Georg Carle. Application of forecasting techniques and control charts for traffic anomaly detection. In *Proc. of ITC Specialist Seminar on Network Usage and Traffic*, Berlin, Germany, October 2008.

[MC08b] Gerhard Münz and Georg Carle. Distributed network analysis using TOPAS and Wireshark. In *Proc. of IEEE Workshop on End-to-End Monitoring Techniques and Services (E2EMon) 2008*, Salvador-Bahia, Brazil, April 2008.

[MCA10] Gerhard Münz, Benoit Claise, and Paul Aitken. Configuration Data Model for IPFIX and PSAMP. Internet-Draft (work in progress), draft-ietf-ipfix-configuration-model-05.txt, March 2010.

[McH04] John McHugh. Sets, bags, and rock and roll: Analyzing large data sets of network data. In *Proc. of European Symposium on Research in Computer Security (ESORICS) 2004*, Sophia Antipolis, France, September 2004.

[MDBC10] Gerhard Münz, Hui Dai, Lothar Braun, and Georg Carle. TCP traffic classification using Markov models. In *Proc. of Traffic Monitoring and Analysis Workshop (TMA) 2010*, Zurich, Switzerland, April 2010.

[MFC$^+$05] Gerhard Münz, Ali Fessi, Georg Carle, Oliver Paul, Dusan Gabrijelcic, Yannick Carlinet, Sherif Yusuf, Morris Sloman, Vrizlynn Thing, Jan van Lunteren, Patricia Sagmeister, and Gero Dittmann. Diadem Firewall: Web server overload attack detection and response. In *Proc. of Broadband Europe (BBEurope) 2005*, Bordeaux, France, December 2005.

[MK00] Keith McCloghrie and Frank Kastenholz. The Interfaces Group MIB. RFC 2863 (Draft Standard), June 2000.

[MKHS10] Tatsuya Mori, Ryoichi Kawahara, Haruhisa Hasegawa, and Shinsuke Shimogawa. Characterizing traffic flows originating from large-scale video sharing services. In *Proc. of 2nd International Workshop on Traffic Monitoring and Analysis (TMA'10)*, pages 17–31, Zurich, Switzerland, April 2010.

[MLC07] Gerhard Münz, Sa Li, and Georg Carle. Traffic anomaly detection using k-means clustering. In *Proc. of GI/ITG-Workshop Leistungs-, Zuverlässigkeits- und Verlässlichkeitsbewertung von Kommunikationsnetzen und Verteilten Systemen (MMBnet) 2007*, Hamburg, Germany, September 2007.

[Mon05] Douglas C. Montgomery. *Introduction to Statistical Quality Control*. John Wiley & Sons, 5th edition, 2005.

[MW47] H. B. Mann and D. R. Whitney. On a test of whether one of two random variables is stochastically larger than the other. *Annals of Mathematical Statistics*, 18(1):50–60, 1947.

[MWC07] Gerhard Münz, Nico Weber, and Georg Carle. Signature detection in sampled packets. In *Proc. of IEEE Workshop on Monitoring, Attack Detection and Mitigation (MonAM) 2007*, Toulouse, France, November 2007.

[MyS10] MySQL Homepage. http://www.mysql.com, 2010.

[NfD10] NfDump Homepage. http://nfdump.sourceforge.net/, 2010.

[Odl03] Andrew M. Odlyzko. Internet traffic growth: Sources and implications. In *Proc. of Optical Transmission Systems and Equipment for WDM Networking II*, pages 1–15, Orlando, FL, USA, September 2003.

[Pag54] Ewan S. Page. Continuous inspection schemes. *Biometrika*, 41(1–2):100–115, 1954.

[Pau06] Olivier Paul. Improving web servers focused DoS attacks detection. In *Proc. of IEEE/IST Workshop on Monitoring, Attack Detection and Mitigation (MonAM) 2006*, Tübingen, Germany, September 2006.

[Pax99] Vern Paxson. Bro: A system for detecting network intruders in real-time. *Comuter Networks*, 31(23–24):2435–2463, December 1999.

[PF95] Vern Paxson and Sally Floyd. Wide-area traffic: the failure of Poisson modeling. *IEEE/ACM Transactions on Networking*, 3(3):226–244, June 1995.

[PLR03] Tao Peng, Christopher Leckie, and Kotagiri Ramamohanarao. Detecting reflector attacks by sharing beliefs. In *Proc. of IEEE Global Communications Conference (GLOBECOM) 2003*, December 2003.

[PLR04] Tao Peng, Christopher Leckie, and Kotagiri Ramamohanarao. Proactively detecting distributed denial of service attacks using source IP address monitoring. In *Proc. of IFIP Networking Conference (NETWORKING) 2004*, Athens, Greece, May 2004.

[PPM01] Peter Phaal, Sonia Panchen, and Neil McKee. InMon Corporation's sFlow: A Method for Monitoring Traffic in Switched and Routed Networks. RFC 3176 (Informational), September 2001.

[PR90] Joseph J. Pignatiello and George C. Runger. Comparisons of multivariate CUSUM charts. *Journal of Quality Technology*, 22(3):173–186, 1990.

[Pre02] Randy Presuhn. Version 2 of the Protocol Operations for the Simple Network Management Protocol (SNMP). RFC 3416 (Standard), December 2002.

[Pyt10] Python Homepage. http://www.python.org, 2010.

[QBCM08] Jürgen Quittek, Stewart Bryant, Benoit Claise, and Jeff Meyer. Information Model for IP Flow Information Export. RFC 5102 (Proposed Standard), January 2008.

[QZCZ04] Jürgen Quittek, Tanja Zseby, Benoit Claise, and Sebastian Zander. Requirements for IP Flow Information Export (IPFIX). RFC 3917 (Informational), October 2004.

[RAM96] George C. Runger, Frank .B. Alt, and D.C. Montgomery. Contributors to a multivariate statistical process control chart signal. *Communications in Statistics - Theory and Methods*, 25(10):2203–2213, 1996.

[Ray93] Bonnie K. Ray. Modeling long-memory processes for optimal long-range prediction. *Journal of Time Series Analysis*, 14(5):511–525, September 1993.

[RH78] P.B. Robinson and T.Y. Ho. Average run lengths of geometric moving average charts by numerical methods. *Technometrics*, 20(1):85–93, February 1978.

[Rob59] S.W. Roberts. Control chart tests based on geometric moving averages. *Technometrics*, 1(3):239–250, August 1959.

[Roe99] Martin Roesch. Snort: Lightweight intrusion detection for networks. In *Proc. of USENIX Conference on System Administration (LISA) 1999*, pages 229–238. USENIX Association, November 1999.

[Rou09] Matthew Roughan. On the beneficial impact of strong correlations for anomaly detection. *Stochastic Models*, 25(1):1–27, January 2009.

[RRSD07] Haakon Ringberg, Jennifer Rexford, Augustin Soule, and Christophe Diot. Sensitivity of PCA for traffic anomaly detection. In *Proc. of ACM International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS) 2007*, pages 109–120, San Diego, CA, USA, June 2007.

[RS07] Yacine Rebahi and Dorgham Sisalem. Change-point detection for voice over IP denial of service attacks. In *Proc. of ITG/GI Fachtagung Kommunikation in Verteilten Systemen (KiVS) 2007*, Bern, Switzerland, February 2007.

[SBK08] Marc Ph. Stoecklin, Jean-Yves Le Boudec, and Andreas Kind. A two-layered anomaly detection technique based on multi-modal flow behavior models. In *Proc. of Passive and Active Measurement Conference (PAM) 2008*, pages 212–221, Cleveland, OH, USA, April 2008.

[SBS09] Dominik Schatzmann, Martin Burkhart, and Thrasyvoulos Spyropoulos. Inferring spammers in the network core. In *Proc. of Passive and Active Measurement Conference (PAM) 2009*, Seoul, South Korea, April 2009.

[SCSC03] Mei-Ling Shyu, Shu-Ching Chen, Kanoksri Sarinnapakorn, and LiWu Chang. A novel anomaly detection scheme based on principal component classifier. In *Proc. of IEEE Foundations and New Directions of Data Mining Workshop*, pages 172–179, Melbourne, FL, USA., November 2003.

[Seb84] George A. F. Seber. *Multivariate Observations.* Wiley-Interscience, 1984.

[She31] Walter A. Shewhart. *Economic Control of Quality Manufactured Product.* D.Van Nostrand Reinhold, Princeton, NJ, 1931.

[Shi63] Albert N. Shiryaev. On optimum methods in quickest detection problems. *Theory Probability and Applications*, 8(2):22–46, 1963.

[SP04] Vasilios A. Siris and Fotini Papagalou. Application of anomaly detection algorithms for detecting SYN flooding attacks. In *Proc. of IEEE Global Communications Conference (GLOBECOM) 2004*, Dallas, TX, USA, November 2004.

[SRX$^+$04] Randall Stewart, Michael Ramalho, Qiaobing Xie, Michael Tuexen, and Phillip Conrad. Stream Control Transmission

Protocol (SCTP) Partial Reliability Extension. RFC 3758 (Proposed Standard), May 2004.

[SSRD07] Augustin Soule, Fernando Silveira, Haakon Ringberg, and Christophe Diot. Challenging the supremacy of traffic matrices in anomaly detection. In *Proc. of Internet Measurement Conference (IMC) 2007*, San Diego, CA, USA, October 2007.

[SST05] Augustin Soule, Kavé Salamatian, and Nina Taft. Combining filtering and statistical methods for anomaly detection. In *Proc. of Internet Measurement Conference (IMC) 2005*, Berkeley, CA, USA, October 2005.

[Sta01] William Stallings. *High-Speed Networks and Internets: Performance and Quality of Service.* Prentice Hall, 2nd edition, 2001.

[SVG07] Osman Salem, Sandrine Vaton, and Annie Gravey. An efficient online anomalies detection mechanism for high-speed networks. In *Proc. of IEEE Workshop on Monitoring, Attack Detection and Mitigation (MonAM) 2007*, Toulouse, France, November 2007.

[SY94] Jeremy Smith and Sanjay Yadav. Forecasting costs incurred from unit differencing fractionally integrated processes. *International Journal of Forecasting*, 10(4):507–514, December 1994.

[TB08] Brian Trammell and Elisa Boschi. Bidirectional Flow Export Using IP Flow Information Export (IPFIX). RFC 5103 (Proposed Standard), January 2008.

[TBM$^+$09] Brian Trammell, Elisa Boschi, Lutz Mark, Tanja Zseby, and Arno Wagner. Specification of the IP Flow Information Export (IPFIX) File Format. RFC 5655 (Proposed Standard), October 2009.

[TBSM09] Bernhard Tellenbach, Martin Burkhart, Didier Sornette, and Thomas Maillart. Beyond shannon: Characterizing internet traffic with generalized entropy metrics. In *Proc. of Passive and Active Measurement Conference (PAM) 2009*, Seoul, South Korea, April 2009.

[TJ98] Marina Thottan and Chuanyi Ji. Adaptive thresholding for proactive network problem detection. In *Proc. of IEEE Third International Workshop on Systems Management (SMW) 1998*, Newport, RI, USA, April 1998.

[TRBK06a] Alexander G. Tartakovsky, Boris L. Rozovskii, Rudolf B. Blazek, and Hongjoong Kim. Detection of intrusions in information systems by sequential change-point methods. *Statistical Methodology*, 3(3), July 2006.

[TRBK06b] Alexander G. Tartakovsky, Boris L. Rozovskii, Rudolf B. Blazek, and Hongjoong Kim. A novel approach to detection of intrusions in computer networks via adaptive sequential and batch-sequential change-point detection methods. *IEEE Transactions on Signal Processing*, 54(9), September 2006.

[UB01] Steve Uhlig and Olivier Bonaventure. Understanding the long-term self-similarity of Internet traffic. In *Proc. of Workshop on Quality of Future Internet Services*, pages 286–298, Coimbra, Portugal, September 2001.

[VER10] VERMONT Homepage. http://vermont.berlios.de, 2010.

[Wal00] Steve Waldbusser. Remote Network Monitoring Management Information Base. RFC 2819 (Standard), May 2000.

[Wal06] Steve Waldbusser. Remote Network Monitoring Management Information Base Version 2. RFC 4502 (Draft Standard), May 2006.

[Wir10] Wireshark Homepage. http://www.wireshark.org, 2010.

[Woo03] Jeffrey M. Wooldridge. *Introductory econometrics - a modern approach.* Thomson South-Western, 2nd edition, 2003.

[WP98] Walter Willinger and Vern Paxson. Where mathematics meets the Internet. *Notices of the American Mathematical Society*, 45(8):961–970, September 1998.

[WS05] Qingtao Wu and Zhiqing Shao. Network anomaly detection using time series analysis. In *Proc. of IEEE International Conference on Networking and Services (ICNS'05)*, Papeete, Tahiti, French Polynesia, October 2005.

[WTSW97] Walter Willinger, Murad S. Taqqu, Robert Sherman, and Daniel V. Wilson. Self-similarity through high-variability: statistical analysis of Ethernet LAN traffic at the source level. *IEEE/ACM Transactions on Networking*, 5(1):71–86, February 1997.

[WVZT90] Kyu-Young Whang, Brad T. Vander-Zanden, and Howard M. Taylor. A linear-time probabilistic counting algorithm for database applications. *ACM Transactions on Database Systems*, 15(2):206–229, June 1990.

[WZS02a] Haining Wang, Danlu Zhang, and Kang G. Shin. Detecting SYN flooding attacks. In *Proc. of IEEE Conference on Computer Communications (INFOCOM) 2002*, June 2002.

[WZS02b] Haining Wang, Danlu Zhang, and Kang G. Shin. SYN-dog: Sniffing SYN flooding sources. In *Proc. of International Conference on Distributed Computing Systems (ICDCS) 2002*, Vienna, Austria, July 2002.

[XFAM02] Jun Xu, Jinliang Fan, Mostafa H. Ammar, and Sue B. Moon. Prefix-preserving IP address anonymization: Measurement-based security evaluation and a new cryptography-based scheme. In *Proc. of IEEE International Conference on Network Protocols (ICNP) 2002*, Paris, France, November 2002.

[YVC03] Nong Ye, Sean Vilbert, and Qiang Chen. Computer intrusion detection through EWMA for autocorrelated und uncorrelated data. *IEEE Transactions on Reliability*, 52(1):75–82, March 2003.

[Ziv06] Artur Ziviani. An overview of internet measurements: Fundamentals, techniques, and trends. *African Journal of Information and Communication Technology (AJICT), UTSePress*, 2(1):39–49, March 2006.

[ZMD$^+$09] Tanja Zseby, Maurizio Molina, Nick Duffield, Saverio Niccolini, and Frederic Raspall. Sampling and Filtering Techniques for IP Packet Selection. RFC 5475 (Proposed Standard), March 2009.

ISBN 3-937201-12-2

ISSN 1868-2634 (print)
ISSN 1868-2642 (electronic)