Periodicity detection in network traffic

Master's thesis

by

Jeroen van Splunder

defended on

21 August, 2015

Mathematisch Instituut

Universiteit Leiden

Supervisor
dr. F.M. Spieksma

Performance of Networks and Systems
TNO

Supervisors
dr. ir. H.B. Meeuwissen
T. Attema MSc

TNO/Jeroen van Splunder, 2014–2015
https://www.jvsplunder.nl/
jeroen@vansplunder.net

Summary

This thesis seeks to answer the following question: *How can one detect periodicity in network traffic from netflow records?*

Netflow records, a widely-used format for summarising the 'traffic flows' going in and out of (corporate) networks, can be used by intrusion detection systems to monitor anomalous behaviour in such networks. Detecting periodicity helps to discern between computer-generated traffic and traffic initiated by human behaviour. An intrusion detection system can use this information, along with other features of the netflow data to classify traffic as anomalous or not.

Periodicity is exhibited by many phenomena in a variety of disciplines — e.g., biology, astronomy, computer science — and many studies look at periodicity to better understand the underlying phenomenon which is at work. This makes periodicity detection a very broad subject, with a variety of different techniques, data forms and applications, described in a rich set of literature.

We observed that most works in literature focus on their own method and domain and that an overview is lacking. For this reason, a broad description of periodicity is given in Chapter 1. A classification is introduced and the techniques in the available literature are indexed according to this classification and the domain they are applied to, creating a 'taxonomy of periodicity research' in Chapter 2. To my knowledge, such an overview of periodicity and techniques of detecting periodicity was not available to this date. Therefore, these chapters should be of use to anyone who wishes to study periodic behaviour and is looking for a suitable data representation and periodicity detection technique.

Chapter 3 explains the workings of network traffic and netflow data. We discuss the polling behaviour associated with command and control channel traffic and how this leads to periodicity in network traffic. In Chapter 4, summaries of relevant literature on periodicity detection in network traffic are given.

Subsequently, several methods of detecting periodicity are applied to netflow data from a university network in Chapters 5 to 7. In Chapter 5, an existing method for finding simple polling behaviour from the interarrival times is refined so that it can be applied to a large data set of a whole network. In Chapter 6, a new method is developed to find more complex periodic patterns in the data set, which were not yet described in the literature. Chapter 7 discusses a new method to take into account multi-dimensional information from the network traffic besides

the timing information, such as the number of packets and bytes in a flow and the duration of a flow.

Finally, Chapter 8 gives an overview of the techniques used and their effectiveness, along with advice on implementations and further research.

Acknowledgments

TNO gave me the possibility to write my thesis during an internship. Erik Meeuwissen and Thomas Attema were my daily supervisors. Erik, thanks for being critical and optimistic, always at the right time, and for your talent to see and emphasize the new and interesting things in a large body of work. Thomas, thank you for your critical and rigorous attitude towards the (mathematical) details of my thesis and your enthusiasm and positive attitude w.r.t. my progress. Thanks to manager Dick van Smirren for letting interns truly be a part of the team and for his three-weekly coaching sessions, which I highly valued. Sander de Kievit kindly helped me by processing the data set and providing me with some of his wide expertise on netflows. Jan Sipke van der Veen swiftly supplied a virtual machine to run experiments on. Pieter Venemans gave an engaging internal course to staff and interns on networking and the OSI model. Coffee and lunch breaks, after-hours socializing and the TNO football tournament with fellow interns and staff members made TNO a pleasant environment to work in.

Thanks to Floske Spieksma, who was my supervisor from Leiden University not only for this Master's thesis but also for my Bachelor's thesis. Every meeting we had provided me with new ideas and optimism and we always had a nice chat, also on non-mathematical topics.

The DACS group of Twente University provided the public netflow data set used in this thesis. Anna Sperotto and Rick Hofstede of this group kindly assisted me when I had questions about the data set.

Glossary

Anomaly detection: The detection of events which are unexpected or unwelcome.

Command and control channel: Communication channel over which an attacker sends instructions to computers which have been taken over.

Connection: The sequence of flows from one IP address to another in a certain time window; see Page 19. For TCP connections, see Page 15.

IDS: Intrusion Detection System, software that monitors a computer network for possible intrusions or malicious activity (anomalies).

IETF: Internet Engineering Task Force, a standards organisation for the internet.

IP: Internet Protocol, an IETF standard, the principal protocol for sending packets over the internet.

IP address: Numerical value which identifies a computer or other device in a network.

IPFIX: Internet Protocol Flow Information Export; the IETF standard for flow collection, based on NetFlow version 9.

Malware: Malicious software.

NetFlow: A standard for netflow collection originally introduced in CISCO routers. Different version of the standard exist; version 5 and 9 are common.

(Net)flow: A unidirectional sequence of consecutive IP packets which share common characteristics, usually source and destination IP address and port and protocol. A flow is usually deemed to end after a time-out or after the connection is explicitly closed (in TCP traffic). The term is often used as a shorthand for the *record* of the netflow.

(Net)flow record: The characteristics of a netflow, as recorded by a netflow collector.Often referred to as (net)flow, omitting 'record'.

Packet: The unit of data in IP. A packet consists of a header and payload.

Port: Number used to separate network traffic intended for different applications on the same computer, e.g., TCP port 25 is commonly used by email servers..

TCP: Transmission Control Protocol, an IETF standard, is used for reliable delivery of data over IP.

UDP: User Datagram Protocol, an IETF standard, is used for simple connectionless delivery of data over IP.

Contents

CHAPTER 1

Periodicity in general

1.1. Introduction

Periodicity, which we for now loosely define as the occurring of similar observations in more or less regular intervals, is a property exhibited by many processes that are of interest in a variety of scientific disciplines. Detecting whether or not these processes exhibit periodicity and learning the details of these periodicities can help in understanding the underlying processes.

In this chapter, we wish to highlight the importance of periodicity detection in general and give some examples of different ways in which processes can be periodic. We also discuss a categorisation we made of the different data forms in which periodicity can be studied. This categorisation is used in the next chapter to index the existing literature on periodicity detection techniques. This should make these chapters useful to researchers who wish to find a suitable periodicity detection technique.

1.2. Importance of periodicity

Periodicity detection has successfully been applied in a wide variety of domains.

In studying the behaviour of moving objects, such as animals equipped with GPS, Li et. al. [1] pose that "finding periodic behaviours is essential to understanding object movements."

In astronomy, periodicity detection is used to detect pulsating stars and eclipsing binary stars [2].

In biology, periodicity detection is used to detect regularity in fertility cycles [3] and the effect of circadian rhythms on animal behaviour and physiology [4, Ch.17].

In computer network traffic, periodicity is an indicator of malware command and control traffic [5], network congestion [6] and denial of service attacks [7]. Contrastingly, in traffic from industrial control systems, disruptions of periodicity are an indicator of attacks [8]. The use of periodicity detection to detect intrusions is described in Chapter 3.

1.3. Examples

Example 1.1 (Tides). The Meetnet Vlaamse Banken is a monitoring system for oceanographic and meteorological data of the Belgian coast and the Belgian

part of the North Sea. The water level in Oostende is made available online [9]. The data consists of the water level in cm relative to TAW, $\{253.9, 270.2, \ldots, 302.1\}$. The data starts at 2015-01-06 12:00 noon and each consecutive measurement is five minutes apart from the previous one.

Looking at the graphic representation of a few days of water level data in figure 1.1, it seems that the water level (tide) on any point of a given day is more or less the same as the water level the day before and after. The similarity becomes even stronger if we consider lunar days — lasting about 25 hours — instead. Though not exactly the same, measurement are *more or less* the same after *about* 25 hours and hence we call the water level periodic.

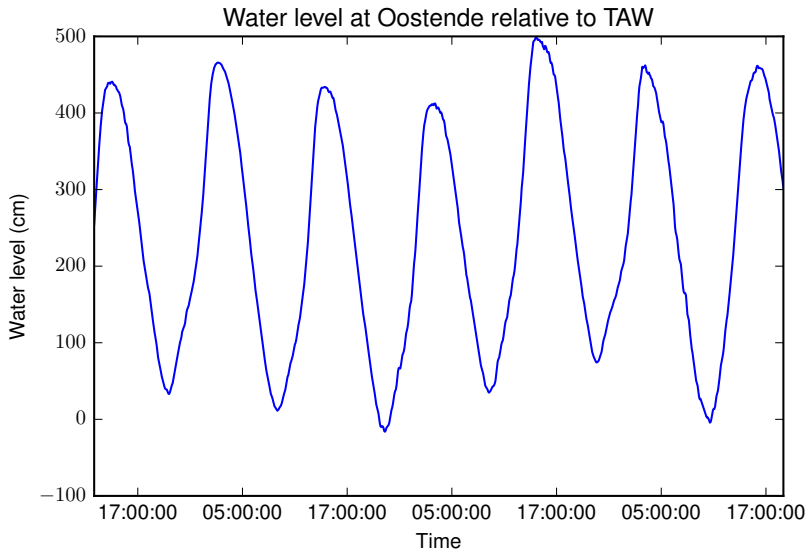

FIGURE 1.1. The tide in Oostende, plotted from a time series of the water level measured with five-minute intervals [9]. The data span 2015-01-06 12:00 noon to 2015-01-09 19:00.

EXAMPLE 1.2 (Network flows). Figure 1.2 shows a sequence of network flows between two IP addresses from the University of Twente data set (see Chapter 3 for more on network flows and this data set). In the figure, the flows are shown consecutively with their starting times. If we divide the flows in groups of size two, the interarrival times within the groups and between the groups is (almost) the same, which is why we label this traffic as being periodic with period 2. Chapter 6 describes the technique that detected this connection.

EXAMPLE 1.3 (DNA sequences). In DNA sequences, micro-satellites are repetitions of a pattern of nucleotides. They are used in forensic identification and in studies of genetic linkage. Finding new micro-satellites is a form of periodicity

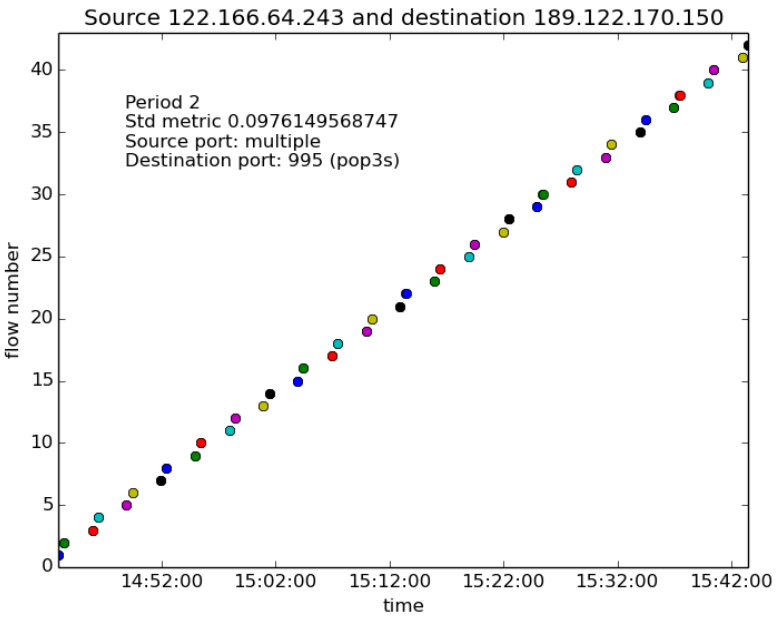

FIGURE 1.2. The network flows in data set 42 of the University of Twente data set from (anonymized) IP address 122.166.64.243 to 189.122.170.150. The port number 995 on the destination side is normally used by email servers (POP3S protocol).

detection. The sequence below comes from a piece of human Y-chromosome and contains the micro-satellite "gata" [10].

```
gacggccagt gaatcgactc ggtaccactt cctaggtttt cttctcgagt tgttatggtt
ttaggtctaa catttaagtc tttaatctat cttgaattaa tagattcaag gtgatagata
tacagataga tagatacata ggtggagaca gatagatgat aaatagaaga tagatagata
gatagataga tagatagata gatagataga tagaaagtat aagtaaagag atgatgggta
aaagaattcc aagccaggca tggtggatta tgcctgtaat cccaacattt tgggagg
```

1.4. Defining periodicity

As noted before, periodicity is the property that similar observations occur in more or less regular intervals. The (approximate or mean) interval length, or the number of observations in such an interval, is called the period.

The intervals are defined on some natural ordering, mostly time. In other cases, such as when describing nucleotide types in DNA, the ordering is positional. For simplicity, we will often refer to orderings in general as 'time'.

The observations may be measurements made from something that could also have been observed at another time, such as liver glycogen levels, or observations triggered by the occurrence of an event, such as the number of bytes in a packet,

measured when the packet passes through a computer network. In the latter case, the timing of the observation gives extra information and we refer to such event-triggered observations as events.

Observations may be periodic only in a part of the data, they may not occur every period or there may be missing observations due to errors in measurement. This has to be taken into account when deciding what one considers 'regular'.

Both in the wording of 'similar' and 'more or less regular' in the definition above, there is a deliberate lack of preciseness. Depending on the domain, existing theory and practical experience, one has to determine what one wishes to see as similar and regular. It is also important to think over (in advance) how one wishes to collect data and how the underlying process that is being researched is or might be generating periodic data.

1.5. Data collection

When detecting periodicity, the data comes in the form of ordered observations. Time, if it is recorded, has a special place, so we denote t_i for the time of observation i, and a_i for (a vector of) other measurements. Remark again, that 'time' is used to denote an ordering in general, which could also be based on spatial position or something else.

Time may be explicitly recorded, implicitly available in the data or not available at all. We will categorise the different data types according to this property and give them names.

If a sequence comes from the regular sampling of a process or an aggregation of events over fixed time bins, the sequence is called a *time series*. In this case, time is implicitly present in the data, such as in Example 1.1.

Time is explicitly recorded in sequences of the form $\{(t_1, a_1), \ldots, (t_n, a_n)\}$. Here, t_i denotes the time of the measurement and a_i is a vector representing the values of the other data measured. If the t_i denote (the start of) an event, such a sequence is called an *event sequence*, because the time information also tells us that something occurred at that time. If the t_i merely denote that a measurement was made at that specific time (but the time itself is not in any way special), the sequence is called an *irregular time series*. This is often the case if measurements are missing or data from different sources are combined.

In Example 1.2, the time t_i recorded is the start time of a flow and the other information a_i consists of the duration of the flows. Because the t_i indicate an event — the start of a flow — the example is an event sequence.

If only the time of the event and no further details are stored, the event sequence is called a *point sequence* $\{t_1, \ldots, t_n\}$. For example, the 'Std metric' listed in Figure 1.2 from Example 1.2 is calculated based on the point sequence of flow start times.

A sequence in which time is not recorded but where the values do contain an ordering, a *value sequence*, is of the form $\{a_1, \ldots, a_n\}$, where a_i is a vector of some

data values. Often, the data a_i come from a finite set of possible values, symbols. In such case, a value sequence may be called a *symbol sequence*.

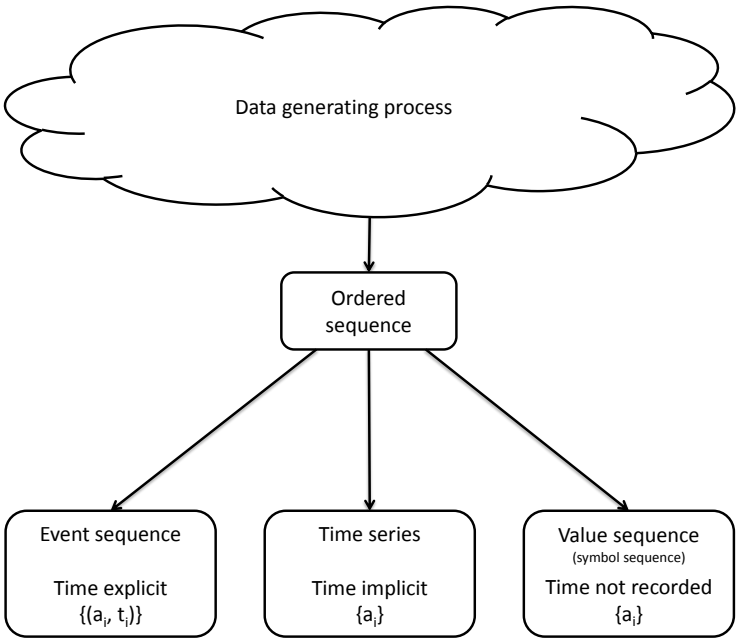

FIGURE 1.3. Measurement on a data generating process yield an ordered sequence. If time is explicitly recorded, the sequence is an event sequence or irregular time series $\{(a_i, t_i)\}$. If time is implicitly recorded because of equal spacing, the sequence is a time series $\{a_i\}$. If time is not recorded, it is a value sequence $\{a_i\}$, or — when the set of possible values is finite — a symbol sequence.

1.6. Measuring periodicity

In a process we consider periodic, the recorded time in-between events and the other data measured may vary somewhat because of (1) random variations in the data generating process; (2) we do not directly measure the process that is periodic, e.g. because of a variable delay when sending packets over the internet; (3) because of measurement errors.

Both for the variations in time and data, one requires domain knowledge to know or define what 'similar' and 'periodic' mean and how much variation is acceptable. In general, some sort of metric has to be defined which gives a score to the data, defining 'how periodic' it is w.r.t. some pattern. Then, a suitable algorithm can be deployed to find these patterns.

To increase ease of computation or because of limitations in measurement capabilities, some data may be neglected or aggregated. For example, one might ignore

the measurements of an event sequence, leaving a point sequence and thus only analysing the time between events. Dividing the time horizon into equally large time bins and aggregating measurements for events in the same time bin creates a time series. Values can be discretised to form a symbol sequence. Depending on the domain, it can be wise to transform the raw data to such an easier to compute data form or to set up the measurement set-up such that it collects these forms of sequences.

CHAPTER 2

General taxonomy of periodicity literature

Much research has been done on finding periodicity and the associated patterns. Most works in the literature describe only the specific methods that were used for solving a specific problem in a specific domain or discipline. Four more general works [4, 11, 12, 13] exist. Chapter 14 of [4] discusses a variety of ways to fit time series to periodic functions, all in the context of biology. In [11] and [12], short summaries of several articles on finding partial periodic patterns (see Section 2.4) in symbol sequences are given. The overview article [13] describes finding exact matches in a symbol sequence and finding patterns that cover a symbol sequence. Using the terminology introduced in Section 1.5, the existing literature on periodicity detection is categorised below.

2.1. Event sequences

2.1.1. Combinations of techniques (reduction). A reductionist approach is to reduce the data of an event sequence to separate information about the time between events (point sequence) and the features of the events themselves (value or symbol sequence).

Ma and Hellerstein[14] use a combination of a point sequences technique to find periods and a symbol sequences technique to discover temporal patterns. They implement both a period-first and a pattern-first algorithm and apply them to data of utilisation rates of computer networks and servers. In the period-first method, they split up events of an event sequence in different types and analyse the point sequences for each type separately as in Section 2.2. For the periods thus found, they apply symbol sequence techniques. In the pattern-first algorithm, a pattern is searched for with symbol sequence techniques and afterwards it is verified that the associated times are also periodic with a point sequence technique.

2.2. Point sequences

In point sequences, any information about events apart from the time they take place is ignored. Thus, we try to detect periodicity by looking at statistical features of the time information associated with events, such as distribution or moments.

Seaton [3] performs a linear regression on rank versus time. First, candidate periods are determined by the researcher. Then, for each candidate period, every

observation is given a rank, grosso modo by dividing the time of the observation by the candidate period and rounding. A linear regression is performed and the coefficient of periodicity is defined as the standard deviation from regression divided by the period length. The candidate period with lowest coefficient of periodicity is taken to be the period of the data, provided its coefficient is low enough to warrant few observations being far from the regression line. The method is applied to oestrus cycles in animals.

Hubballi and Goyal [15] note that periodic events have similar interarrival times, causing the variance of these interarrival times to be low. This is used to analyse flows of computer network traffic; see Section 4.1.1.

Ma and Hellerstein [14] consider the number of events that have an interarrival time within some margin δ of an interarrival time τ. This is then compared with a randomly generated sequence of the same length, and a chi-squared test with 95% confidence level is employed to determine whether τ is a possible period.

McPherson and Ortega [16, 17] use the fact that packet arrivals in a large-scale computer network can be modelled as a renewal process. When packets arrive periodically due to a DoS-attack or a bottleneck in the network, the pdf of an arrival occurring will be different. This is used to perform a Chi-squared test. For a detailed description of these articles, see Section 4.1.4.

2.3. Time series and value sequences

2.3.1. Information-theoretic metric. Huijse et.al. [2] use an information-theoretic metric to define a distance based on time and magnitude differences to investigate pulsating stars with an irregular time series. The null hypothesis — no periodicity — is then simulated and compared with the measurements.

2.3.2. Fitting to a periodic function. The following definition of a periodic function can be found in most Calculus books.

DEFINITION 2.1. *A function $f : \mathbb{R} \to \mathbb{R}$ is (strictly) periodic if there is a period $p \in \mathbb{R}$ such that $f(x + p) = f(x)$ for all $x \in \mathbb{R}$.*

A time series consists of measurements $a_1, \ldots, a_n$ for a finite set of times $t_1, \ldots, t_n$. (The times may be implicit if the time series is regular, or we can take the ordering in the sequence as 'time' for a value sequence.) If these values fit very well to a strictly periodic function, then we can say that the process is periodic. Since every periodic function can be written as a Fourier series, it is natural to try to fit a finite sum of sine and cosine functions to the event sequence. Chapter 14 in [4] describes the statistical methods to find and test the statistical significance of such a fit, along with many examples from the field of biology.

2.3.3. Spectral analysis. For regular time series and for value sequences, it is possible to transform the data to the spectral domain using discrete Fourier transforms or wavelets. By looking at frequencies with high energies, periods

can then be found, e.g. using an experimental threshold or a hypothesis test. Several examples of articles using this approach in network traffic are described in Chapter 4.

2.3.4. Segmentation. In [18], Indyk et al. define two definitions of periodicity for time series (they apply equally well to value sequences). Given a time series V of length N and a candidate period T, cut V into segments $V(T) = \{v_1, v_2, \ldots, v_n\}$ where each of the segments v_i is of length T, possibly ignoring the last values of V if N does not evenly divide into T. For a distance function d which defines the distances $d(v_i, v_j)$ between any pair of segments, define the distance between segment i and $V(T)$ as

$$D(V(T), v_i) = \sum_{j=1}^{n} d(v_i, v_j). \tag{2.1}$$

A natural choice for the distance function d is the Euclidean distance. Given the distance $D(V(T), v_i)$ and an interval $[l, u]$ in which possible values of T lie (or to which one wishes to constrain the search for such periods, e.g. for reasons of performance), the relaxed period and average trend can be defined.

DEFINITION 2.2 (Relaxed period). *Given a vector V and integers $l \leq u$, the relaxed period of V in the range $[l, u]$ is the integer T, $l \leq T \leq u$, for which $D(V(T), v_1)$ is minimised.*

DEFINITION 2.3 (Average trend). *Given a vector V and integers $l \leq u$, the average trend of V in the range $[l, u]$ is the segment v_j such that $D(V(T), v_j)$ is minimal for $l \leq T \leq u$.*

In the definition of the relaxed period, special importance is (arbitrarily) given to the first segment of the vector, which is compared to the other segments. In the definition of the average trend, a segment of the vector is found which best represents the other segments (has low distance to them). This comes at the cost of extra computational complexity, since one needs to compare not only the distance of the first segment all of the segments, but also repeat this procedure for all segments.

In [19], a time series is segmented into length T. The average values in a segment are then computed and for each segment, the correlation with this average segment is calculated. The average of these correlation scores is then the score associated with the segmentation of length T. The T with the lowest score is defined to be the period.

Segmentation is further discussed in Chapter 7.

2.4. Symbol sequences

A symbol sequence is of the form $S = \{a_1, \ldots, a_n\}$, consisting of letters a_i from a finite set of letters A, the alphabet. A strict definition of periodicity is often formulated similar to definition 2.1.

DEFINITION 2.4. *A sequence is* strictly periodic *if there is a* $p \in \mathbb{N}$ *for which* $e_i = e_{i+p}$ *for all* $1 \leq i \leq n-p$. *The sequence* $\{e_1, \ldots, e_{p-1}\}$ *is called a pattern and we say it is periodic with period* p.

EXAMPLE 2.1. In the sequence $S = abcabcabcabcabcabcab$, the pattern abc is repeated. The sequence is strictly periodic with period 3; integer multiples of 3 are also periods of S.

The overview article [13] discusses several questions related to finding exact occurrences of a pattern within a sequence, or finding a pattern with which a sequence can be covered. For more flexibility, one might like to consider sequences which have recurring patterns but where some letters are replaced — e.g., because of noise or errors in transmission. The following example was generated from example 2.1 by randomly replacing (with $p = \frac{1}{4}$) some of the letters by a, b or c (uniformly).

EXAMPLE 2.2. The sequence $S = abcabcabaabcbacabcab$ seems to consist of repetitions of the pattern abc with period 3, though sometimes one of the letters was changed.

One way to determine whether such a sequence is still periodic is by cutting up the sequence in groups of 3 and comparing these groups with the pattern abc. One might also be interested in patterns like $a*c$, where the $*$ is a wildcard, which matches any single character in the alphabet. The concepts of *periodic patterns* and *regional periodic patterns* cater to this interest.

DEFINITION 2.5. *A* pattern s *is a non-empty sequence* $s = \{s_i\}_{i=1}^{m}$ *over* $A \cup \{*\}$, *where* $*$ *is a wildcard which matches any single character in the alphabet* A.

In order to determine if a sequence S is periodic, a period p is assumed and S cut up in $C = \lfloor n/p \rfloor$ *periodic segments* $E_i = \{e_{ip+1}, e_{ip+2}, \ldots, e_{ip+p}\}$ (the remaining part of the sequence after C periods is ignored). A periodic segment E_i is said to *match* the pattern s, and $\text{match}(s, E_i) = 1$, if for all $j = 1, \ldots, p$, s_j is the wildcard $*$ or s_j equals the j-th letter of E_i; otherwise, $\text{match}(s, E_i) = 0$. Furthermore, the *frequency* of matches $\text{freq}(s, S)$ and the relative frequency, or *confidence level*, $\text{conf}(s, S)$, are defined as

$$\text{freq}(s, S) = \sum_{i=0}^{C-1} \text{match}(s, E_i), \tag{2.2}$$

$$\text{conf}(s, S) = \frac{\text{freq}(s, S)}{C}. \tag{2.3}$$

If the frequency and confidence level of matches exceed certain predetermined thresholds, the sequence is considered to be periodic.

DEFINITION 2.6. *Let a sequence S, a pattern s and thresholds* minconf *and* minfreq *be given and suppose that*

$$\text{conf}(s, S) \geq \text{minconf}, \tag{2.4}$$

$$\text{freq}(S, s) \geq \text{minfreq}\,. \tag{2.5}$$

Then s is a full periodic pattern *of S if it does not contain the wildcard $*$. Otherwise, if s contains one or more wildcards, s is a* partial periodic pattern *of S.*

For $s = abc$, the sequence S in example 2.2 has 4 matches in 6 periods. This means that for values of minconf $\leq \frac{2}{3}$ and minfreq ≤ 4, s is considered a full periodic pattern of S.

A good introduction to these concepts and and an algorithm for finding (partial) periodic patterns are given in [20]. In this algorithm it is assumed that the period is known a priori, or that a user-defined set of candidate periods or a lower and upper bound are given, after which computations are done for each of those possible periods. In [21], a method is introduced to filter the search space of all possible periods, and only apply the algorithm from [20] to the periods in which it is likely that (partial) periodic patterns might be found. Another approach [22] efficiently computes an estimate of the confidence level for all periods. In [23], an efficient way to compute (partial) periodic patterns for all periods at once is described.

Instead of using the confidence of a pattern, as above, it is possible to use another metric to give a score to partial periodic patterns. In [24], patterns are given a score based on their information entropy, favouring 'unlikely' patterns.

CHAPTER 3

The need for periodicity detection in network traffic

3.1. Purpose of this chapter

In Chapters 1 and 2, we showed how periodicity is of interest in a variety of domains and that there are many different approaches to detecting periodicity. This chapter introduces the domain of network traffic and the role of periodicity therein. Section 3.2 shows how periodicity is of interest in network traffic because of the occurrence of malware in computer networks. In Section 3.3, some background information is given on networking protocols. Section 3.4 describes the netflow data format for describing network traffic and Section 3.5 the data set of netflow data which we will be using in Chapters 5 to 8. In Section 3.6, we state the problem we wish to solve.

3.2. Command and control channels

It is a concern that in any institution, computer systems may be infected with malware under the control of an outside attacker, such as botnets and advanced persistent threats [25]. Such malware poses risks to institutions as services can be disrupted and confidential information may be extracted by the perpetrator behind the malware. One way of finding infected computers is by analysing the network traffic between the internal network of the institution and the outside world.

On the internet layer, the connection between an internal network and the rest of the internet is usually made at one point, making it possible to observe the in- and outgoing traffic at this detection point. When malware is controlled by an outside attacker, it will have to communicate with the attacker, e.g., to receive commands or send back gathered information. If these communications are passed over the internet, they will pass the detection point and can be detected.

The connection between a computer infected by malware and the perpetrator behind the attack is known as a command and control channel [25]. A perpetrator uses such a channel to issue commands to the malware. Conversely, the malware uses the channel to accept orders and to indicate to the attacker that it is (still) controlling a computer — since the malware may have redistributed to new computers on its own or have been removed. The results of issued commands may also be sent over the command and control channel, or over a separate channel.

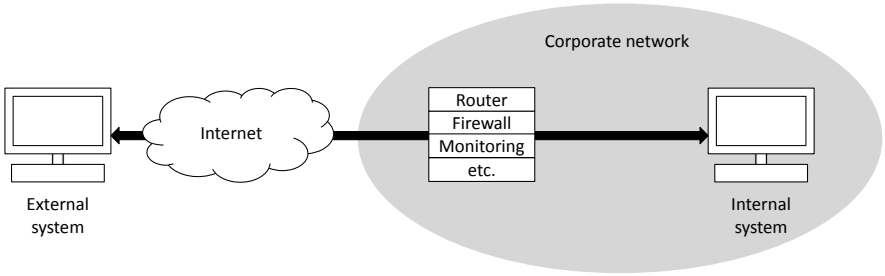

FIGURE 3.1. In a typical corporate network, all connections to and from the internet pass one central router. At this point, a firewall and monitoring applications may also be deployed.

Firewall rules and network address translation typically prevent computers on the internet to open connections with computers in the internal network, unless they are specifically marked as servers. Therefore, connections have to be initiated from within the internal network by the malware. If the command and control channel is an open channel, over which commands can be issued by the attacker in real-time, there has to be regular communication over the channel, or else the connection will be closed off by the firewall after some time. If the command and control channel is a passive channel, where the malware queries an outside resource (such as a website) to see if the attacker has posted new commands to execute, the malware will have to regularly check this resource in order to be informed of new orders.

As seen in traffic analysis in [26], periodic network traffic patterns occur because of command and control channel traffic and these patterns can be used to detect malware. More sophisticated malware might have more erratic polling behaviour, trying to mask as normal traffic. However, without frequent polling the connection between attacker and malware will get closed down, reducing the possibilities of the attacker. Thus, finding these polling behaviours can be of great help in detecting malware and limiting its impact.

We will develop and apply periodicity detection techniques to find polling behaviour in network traffic. This behaviour is common in malware, as discussed above, but also in computer-generated traffic from applications with legitimate uses, such as peer-to-peer applications and e-mail clients [26]. In contrast, network traffic generated by human behaviour is unlikely to exhibit strong periodicity at small timescales. Hence, a metric of periodicity may help to filter traffic which might be malign from traffic which is probably not. This is relevant, since the usefulness of intrusion detection systems is greatly influenced by the false positive rate [27].

3.3. Networking preliminaries: IP, TCP and UDP

Traffic over the internet is sent using the Internet Protocol (IP), predominantly version 4 and to a lesser extent version 6. Computers are assigned a unique number, an IP address, which is used to route packets from one computer to another. The packets sent over IP consist of a header containing information on the source and destination of the packet and the payload, containing data. There are several ways to send messages using IP, called protocols, of which TCP and UDP are the most widely-used.

UDP simply sends the message to the receiver without any way for the sender of knowing whether the message arrived. This is used in applications where timely arrival is important, like in voice over IP. Resending a packet when it is lost is not useful in this case, because the conversation will have moved on by the time the resent packet arrives.

TCP is used when reliable transmission of the data is important. In TCP, computers set up a *TCP connection* where each packet is given a sequence number. Once a packet is correctly received, the receiver acknowledges the delivery by sending back a TCP packet where it indicates the sequence number of the next packet it is expecting. In this way, the sender notices when a packet goes missing and can resend the message.

Both TCP and UDP use the concept of a port to separate different communications. A port is a number which allows the host to associate packets that specify this port to a specific service on the host. Many applications have standard ports, such as TCP port 80 for HTTP (web servers), though this is largely a convention. It is possible for applications to use non-standard ports and this is done in practice by malware [28, 25].

Flags are used in TCP to send information regarding the status of the TCP connection. When a computer wants to open a TCP connection with another computer, it sends a TCP packet with the SYN flag set and an initial sequence number. If the computer on the other sides accepts the connection, it will send a TCP packet with the ACK flag set and the sequence number given by the initiator plus one, acknowledging correct reception of the first packet. Often, the receiver also wishes to relay data to the initiator and it will also choose a sequence number and send this together with a SYN flag. Once this packet has been acknowledged by the initiator, the communication is open in both ways. For each subsequent packet sent the sequence number is raised by one and for each packet correctly received, a message with an ACK flag and sequence number $x + 1$ is sent if all packets up to and including x have correctly been received. When acknowledgement are not timely received, packets are sent once more.

The first ACK and SYN of the receiver side are often sent together, as in figure 3.2. This way of opening a connection in two ways in three steps is called the three-way handshake.

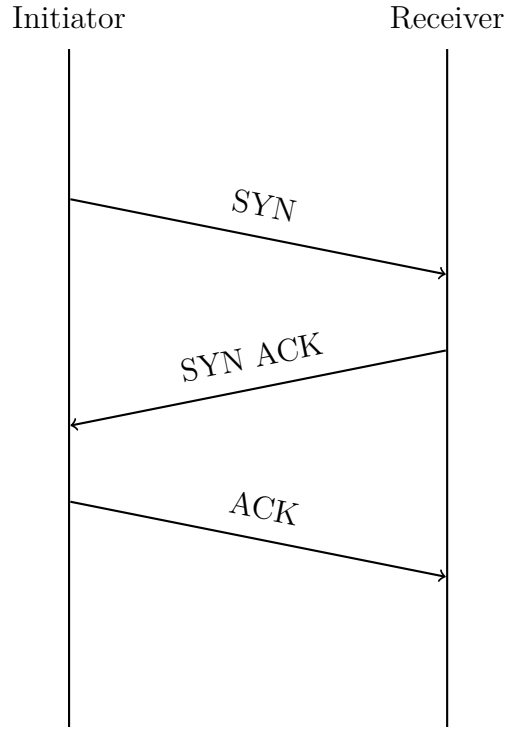

FIGURE 3.2. In this diagram of a three-way handshake, the arrows indicate packets sent and time increases from top to bottom. The initiator sends a packet with the SYN flag set and an initial sequence number because it wishes to open a connection to the receiver. The receiver acknowledges that it has correctly received the message by sending a packet with the ACK flag set and the initial sequence number given by the initiator plus one. In the same packet, it also sets the SYN flag with its own initial sequence number to open up the connection from receiver to initiator. When the initiator receives this packet, it knows the connection is open and it can start sending data to the receiver. Finally, the initiator acknowledges the receipt of the packet from the receiver with an ACK flag and the initial sequence number of the receiver plus one. The connection is now open in both ways.

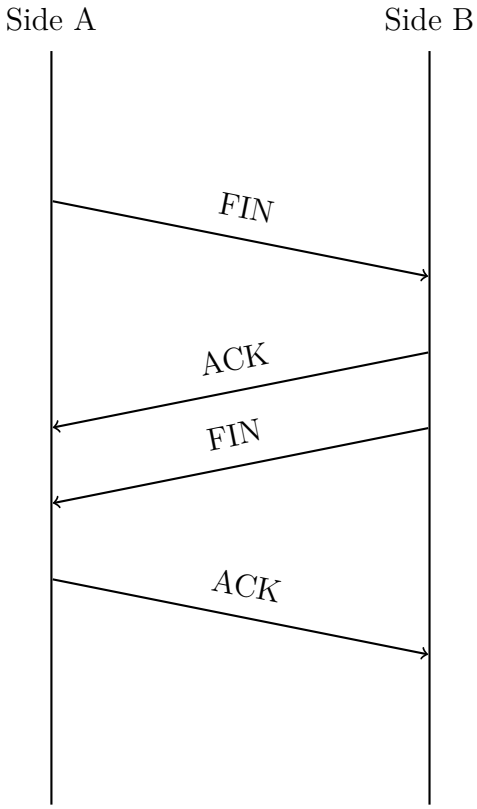

FIGURE 3.3. In this diagram of the closing of a TCP connection, the arrows indicate packets sent and time is shown increasing from top to bottom. Side A wishes to close the connection by sending a packet with the FIN flag. Once it has received confirmation that this packet came through because of the ACK flag set on the packet from side B, the connection is closed. The connection from side B to side A is still open until side B sends a packet with a FIN flag and receives an acknowledgement.

Closing of a TCP connection is done in a similar way to opening, but uses the FIN flag. After a FIN flag has been acknowledged by the other side, the connection is considered closed in that direction. Often, the computer receiving a FIN packet will also want to stop sending data and also sends out a FIN packet, like in figure 3.3.

The RST flag indicates a request for a reset, ending the connection and starting a new one. It is sent by a host when it does not know what to do with the packets it receives, e.g., because something in the connection went wrong or because there is no service running on the specified port.

3.4. Netflow records

In this thesis, we will be looking at netflow records, a summarising technique for the 'flows' of network traffic which are observed. A flow consist of consecutive packets sharing properties, usually the same source and destination IP address and port. For such flows, aggregate statistics on the constituent packets are stored by a netflow collector in netflow records, such as the start and end time of the flow, the number of packets and the number of bytes — an example is given in Table 3.1. Netflow collectors start writing a netflow record when a packet from a source to a destination is observed for which no netflow record is open yet. The netflow record is finished when no new packets have been received for some configurable time (usually around five minutes) or when FIN and ACK signals indicating the closing of a TCP connection are received. There are several different standards for recording netflow records, most notably Cisco NetFlow version 5 and 9 and the IETF standard IPFIX. The differences are small and not relevant to our use case. Similarly, since the difference between a netflow and the associated netflow record is subtle and not very important in our use cases, we will use the terms interchangeably.

As the amount of traffic going through the connection between an internal network and the internet can be huge, even temporarily storing and analyse it is very costly. Netflow data has the advantage that it takes up only 0.2% of the space of the original traffic [30]. Furthermore, in contrast with so-called deep packet inspection, encrypted traffic can be handled and there are less privacy concerns as the content of the traffic is not available. Netflow collection is a standard feature of routers already used in many networks, so that it is simple to set up [31].

3.5. Data set and pre-processing steps

A data set of anonymized netflows from the campus network of the University of Twente [32] is made available online[1]. We will carry out our experiments on this data set because it is publicly available and contains real-life traffic of a large computer network. The data consist of raw dumps of the information sent by a

[1] `http://www.simpleweb.org/wiki/Traces`

TABLE 3.1. Example of the information found in a netflow record. The table is derived from the output of the Nfdump [29] program for a netflow record from the data set described in Section 3.5.

Field	Value
Flow start	2007-07-28 16:40:44.182
Duration (seconds)	0.704
Protocol	TCP
Source IP address	183.201.100.222
Source port	50502
Destination IP address	122.166.69.120
Destination port	6969
Flags	None
Packets	6
Bytes	553
Packets	1

Cisco NetFlow version 5 netflow meter, aggregated per hour. A total of 184 files contain almost 50 GB of compressed netflow records gathered from the campus network between 2007-07-26 and 2007-08-03.

As an example of the size of the data, we take a closer look at file 42. This file contains data on 9,163,642 flows between 584,729 different pairs of IP addresses in the time interval from 15:40 on 2007-07-28 to 16:40 o'clock. (Note that pairs may be counted twice, as flows are uni-directional.) Only 109,585 of these pairs have 10 flows or more.

The data was pre-processed to remove UDP traffic and traffic that was not acknowledged with an ACK flag, or explicitly rejected with a RST flag — indicating an error with the connection, or traffic from a port scanner which tries to connect to many computers indiscriminately. Next, the flows with the same source and destination IP address are grouped to form what we call a *connection*.

DEFINITION 3.1 (Connection). *The* connection *from host A to host B in time window $W = [l, u]$ consists of the flows $f_1, \ldots, f_n$ from A to B with starting times $t_1, \ldots, t_n \in [l, u]$. For convenience, we assume that the flows in a connection are ordered such that $l \leq t_1 \leq t_2 \leq \cdots \leq t_n \leq u$.*

Besides the information on the source and destination port, start and end time, number of octets and bytes which we copy from the flow records, we also include the duration of each flow and the interarrival time of the start of subsequent flows. An example of a connection is given in Table 3.2. Connections with fewer than 10 flows are disregarded.

TABLE 3.2. Connection from 1.197.241.52 to 122.166.253.19. Each row presents a flow, for which the source and destination port (SP and DP), start time, end time, interarrival time (between subsequent flows), duration of the flow and the number of bytes and packets in the flow are given.

SP	DP	Start time	End time	Interarrival (s)	Duration (s)	Bytes	Packets
80	46917	2007-07-28 14:58:28.423000	2007-07-28 14:58:29.255000	—	0.832	36730	30
80	46918	2007-07-28 14:58:29.320000	2007-07-28 14:58:29.384000	0.897	0.064	2823	3
80	46920	2007-07-28 14:58:29.384000	2007-07-28 14:58:29.448000	0.064	0.064	4556	7
80	46919	2007-07-28 14:58:29.384000	2007-07-28 14:58:29.384000	0.0	0.0	3832	4
80	46921	2007-07-28 14:58:29.384000	2007-07-28 14:58:29.384000	0.0	0.0	3809	4
80	46918	2007-07-28 14:58:51.656000	2007-07-28 14:58:51.656000	22.272	0.0	9861	7
80	46919	2007-07-28 14:58:51.721000	2007-07-28 14:58:51.721000	0.065	0.0	15592	11
80	46921	2007-07-28 14:58:54.279000	2007-07-28 14:58:54.407000	2.558	0.128	15063	12
80	46918	2007-07-28 14:58:54.471000	2007-07-28 14:58:54.471000	0.192	0.0	15592	11
80	46919	2007-07-28 14:58:54.791000	2007-07-28 14:58:54.791000	0.32	0.0	4496	6
80	46921	2007-07-28 15:00:20.808000	2007-07-28 15:00:21	86.017	0.192	9545	8
80	46918	2007-07-28 15:00:21.062000	2007-07-28 15:00:21.062000	0.254	0.0	15592	11
80	46921	2007-07-28 15:00:26.182000	2007-07-28 15:00:27.270000	5.12	1.088	21056	18
80	46918	2007-07-28 15:00:26.371000	2007-07-28 15:00:26.371000	0.189	0.0	15592	11
80	46918	2007-07-28 15:02:25.988000	2007-07-28 15:02:25.988000	119.617	0.0	104	2

3.6. Problem statement

A priori, it is unknown whether there are periodic connections observable in our data set, so our first goal is to find such connections, if any exist. Our data set consists of netflow records and as such is a summary of the real traffic observed in the network. Since the dataset does not come with information on the contents of the traffic and the IP addresses are anonymized, the traffic in any connections we find may come from all kinds of processes. Our main goal is to develop periodicity detection techniques for netflow data that allow us to efficiently find the connections which exhibit periodicity, without being swamped by false positives.

CHAPTER 4

Literature on periodicity in network traffic

This chapter describes the literature in which periodicity detection is applied to network traffic. The work of [15] (Section 4.1.1) is applied in Chapter 5 and the method to find a candidate period by [33] (Section 4.2.1) is used in Chapter 6. The description of the other articles serves to give an overview of the current state of periodicity detection in network traffic.

4.1. Based on inter-arrival times

4.1.1. Hubballi and Goyal (2013) [15]. Hubballi and Goyal determine periodicity in network traffic by considering the sample standard deviation of the time between successive flows. They choose one host $\mathcal{H}$ and monitor the flows from and to this host (having $\mathcal{H}$ as source IP address or as destination IP address). For every distinct pair $(x, \mathcal{H})$ or $(\mathcal{H}, x)$ of source IP address and destination IP address, the times between successive flows, called 'DiffTimes', are observed. Note that the pairs are ordered – the pair with source x and destination $\mathcal{H}$ is distinct from the pair with source $\mathcal{H}$ and destination x. If the sample standard deviation of the DiffTimes is below a predefined threshold, the communication is considered to be periodic.

The communications of the host $\mathcal{H}$ are monitored using the Tcpdump utility [34], which monitors network traffic on a packet level. The authors use the standard definition of a flow as a set of packets belonging together because of shared properties, starting with the arrival of a first packet. They do not go into detail on how they classify packets as belonging to the same flow, but we may assume that, like in netflow meters (Section 3.4), flows end after a timeout or — in the case of a TCP connection — after a packet with the FIN or RST flag was observed.

Instead of gathering all data and then computing the sample standard deviation, the authors only keep track of a summary of the netflow data, which they call a FlowSummary. This is done to reduce the storage needed as they deem the network traffic to be bulky. A FlowSummary consists of source and destination IP address, linear sum of DiffTimes LS, squared sum of DiffTimes SS, number of flows M and the timestamp of the start of the last observed flow t. With every new flow, the time difference with the previous flow is added to LS, its square to SS, M is increased by one and the new timestamp takes the place of t. The sample standard deviation can be calculated with the information in the FlowSummary

by equation 4.1.

$$SD = \sqrt{\frac{1}{M-1}\left(SS - \frac{(LS)^2}{M}\right)} \tag{4.1}$$

In the first experiment, a computer connected to the internet is used for normal desktop usage for 5 days while concurrently running a script that connects to a web page every 100 seconds plus a random delay of between 1 and 10 seconds. In the second experiment, a web page is loaded that automatically refreshes periodically, together with normal desktop usage for 7 hours.

Pairs of IP addresses are considered to have periodic communications if at least 10 packets are observed and the sample standard deviation is less than 10 seconds[1]. The article mentions that with hindsight a standard variation threshold of 3 seconds would have been enough to identify periodic behaviour and that the threshold is a parameter that can be fine-tuned by the system administrator.

4.1.2. Bilge et al. (2012) [5]. Disclosure [5] is a detection system for botnet[2] command and control servers based on netflow analysis. The authors classify hosts (IP addresses) as server or as client and for each server collect several features of the netflow data relating to the connections between the server and other hosts. A random forest classifier is trained with training data to distinguish botnet command and control servers from benign servers based on the extracted features.

Some of the features are determined from the flows between a server and a host. For flow sizes: the average, sample standard deviation and distribution of unique flow sizes. For the autocorrelation of the flow size per 300 second bin: the average and sample standard deviation. For the interarrival times of flows: the minimum, maximum, median and sample standard deviation.

Other features are determined for the servers, taking together the flows from all clients to and from this server. These are the number of unmatched flows (flows which do not have a corresponding flow in the other direction) and unspecified statistical features of the number of bytes and the number of clients per hour per server.

The authors note that botnets contact their command and control server periodically and with relatively short intervals, which is listed as the reason for including the features of the number of bytes and clients per hour per server.

Resilience (robustness) is tested by simulating botnets which have a random delay between flows and a random padding (increase) of the flow size. This is done first by choosing a delay from a uniform distribution between 1 minute and 1 hour and in a second experiment by choosing the delay from a Poisson distribution

[1]The unit used for the sample standard deviation is not explicitly mentioned in the article, but the authors confirmed by private communication that it is in seconds.

[2]A botnet is a network of computers infected by malware, all under control by the same person or organisation.

which has a mean that is uniformly sampled between 1 minute and 1 hour. Some of these randomised botnets were added to the training data, which caused the others to be detected. The authors further note that the detection rate of real botnets went up after adding randomised botnets to the training data.

4.1.3. Qiao et al. (2012) [35]. The authors use some properties of search requests of the eMule peer-to-peer software to filter packets with such search requests from network traffic. The timestamps of these packets are stored in an ascending time sequence T. The traffic is marked as anomalous if an ascending subsequence $S \subseteq T$ exists which is periodic. In order to define periodicity, three parameters are used: the minimum length of the periodic sequence $K \in \mathbb{N}$, the adjustment ratio $\alpha \in [0, 1]$ and the identification ratio $\omega \in [0, 1]$. For an ascending subsequence $S = \{t_1, \ldots, t_m\}$ with $t_j \in T$, the difference sequence $\Delta S = \{\Delta t_1, \ldots, \Delta t_{m-1}\}$ is defined by $\Delta t_k = t_{k+1} - t_k$, $1 \leq k \leq m-1$ and its average by $\text{Avg} = \frac{1}{m-1} \sum_{k=1}^{m-1} \Delta t_k$. A subsequence S is marked as periodic if it has at least K elements and the fraction of time differences that fall within the adjustment ratio is at least as big as the identification ratio:

$$\frac{1}{m-1} \sum_{k=1}^{m-1} \mathbb{1}_{\Delta t_k \in [\text{Avg}(1-\alpha), \text{Avg}(1+\alpha)]} \geq \omega. \tag{4.2}$$

Iterating through every subsequence S of length at least K, called the passive match algorithm, has complexity $\mathcal{O}(2^n)$. An approximation algorithm, called active search, of $\mathcal{O}(n^2)$ is given, which iteratively seeks for and adds timestamps that can be added without breaking the requirement of Equation (4.2).

4.1.4. Assumptions on the distribution of interarrival times. Several articles make assumptions on the distribution of the interarrival times under a null hypothesis (no periodicity) and use a hypothesis test to detect periodicity. These are the articles by He et al. (2009) [6], McPherson and Ortega (2009) [16] and McPherson and Ortega (2011) [17].

All articles are about finding bottlenecks in networks links, i.e., the cables linking large networks. A bottleneck leads to periodicity because all packets come in right after one another through the link, at the maximum throughput rate. In [16], it is assumed that the arrival of packets in such a link is a Poisson process (which is debatable, but can be a valid assumption at small timescales for large links, containing traffic from many independent sources [36, Ch. 4.2]). The same author, in [17], assumes that the interarrival times are a renewal process, testing the method again with Poisson processes. In [6], it is assumed that the maximum peak of the interarrival times in the spectral domain has a Gaussian distribution, with different mean and standard deviation for bottleneck and non-bottleneck links. A training set is then used to determine these parameters.

4.2. Symbol sequence

4.2.1. Qiao et al. (2013) [33]. This article describes a method of finding peer to peer botnets with periodicity. Qiao et al. seek for periodic patterns in time series of the number of packets per second sent from an IP address in the internal network. The number of packets sent is classified into five levels a up until e, where a stands for a very low amount of packets and e for a high amount of packets, with each of the categories assigned 20% of the observed values. Thus, a symbol sequence consisting of letters from $\{a, b, c, d, e\}$ emerges.

In order to find a candidate period, the autocorrelation — the correlation of the symbol sequence with itself, shifted over k positions — is calculated for various k. The lowest value k for which the autocorrelation function has a local peak for k, $2k$ and $3k$ is used as a candidate period, to which the method of regional periodic patterns is applied (cf. Section 2.4).

4.3. Spectral analysis on time series

4.3.1. Bartlett, Heidemann and Papadopoulos (2011) [26]. In [26], wavelets are used to transform a time series of the number of new flows per time bin to the spectral domain. The energy in a frequency bin is then compared to a threshold, which depends on the width of the frequency bin.

4.3.2. Barbosa, Sadre and Pras (2012) [8]. SCADA systems (supervisory control and data acquisition), which are used to control and monitor industrial systems, tend to send updates on their status in predetermined intervals. The network traffic coming from such systems is thus very periodic in nature. In contrast with our case, periodic traffic is thus the norm in such systems and any deviation from this periodic behaviour is of interest since it could indicate an anomaly. In [8], the time series of the number of packets received in an interval is measured for a SCADA system. The periodogram of this time series, when plotted, shows clear lines because of the periodic traffic. From such a manual observation, anomalies can be detected by irregularities in the plot. The approach is not automated, but some suggestions are given as to how this might be done.

4.3.3. AsSadhan and Moura (2014) [37]. In [37], a time series of the number of packets per 100 ms time bin is used for a given port on a given host. It is assumed that the number of packets per 100 ms time bin follows a Poisson distribution if the traffic is not periodic. This distribution is approximated by a Gaussian distribution. A hypothesis test is then applied on the largest peak in the periodogram using Walker's large sample test.

4.3.4. Heard, Delanchy and Lawson (2014) [38]. In [38], the number of flows from IP address x to y by time t is observed as a time series $N_{xy}(t)$, with the index $t \in \{0, 1, \ldots, T\}$ in seconds. To do a hypothesis test, the number of newly seen flows in time bin t is compared to the average number of arrivals per time

bin $N_{xy}(T)/T$, yielding a time series $Y_t = (N_{xy}(t) - N_{xy}(t-1)) - N_{xy}(T)/T$. It is assumed that Y_t is of the form $Y_t = \beta\cos(\omega t + \phi) + \epsilon_t$, with $\beta \geq 0$ and $\omega \in (0, \pi)$ constant, ϕ uniform in $(-\pi, \pi]$ and $\{\epsilon_t\}$ a set of uncorrelated random variables with mean 0 and variance σ^2, independent of ϕ. The traffic is deemed not to be periodic if $\beta = 0$, which is used as null hypothesis and tested with Fisher's g-test.

CHAPTER 5

Experiment 1: Detecting periodicity with period 1

5.1. Goal

In the first Experiment, we apply a method from [15], which is known to detect periodic traffic in a set-up with one desktop PC, to our data set of netflow data from a campus network with many computers. The goal is to get acquainted with the data set, find at least some of the periodic traffic in the data set and see how well the method functions on our data set. The larger size of our data set will urge us to make several improvements.

With 'period 1' in the title, we allude to the fact that this method only detects connections when their interarrival times are similar from one flow to the next; it will become clear in this chapter that more sophisticated patterns exist, such as a pattern where a short interarrival time is always followed by a longer interarrival time and vice versa. We will say that the first type of pattern has period 1. The discovery of the second type of patterns in this chapter is the reason for the development of a detection technique for periods larger than 1 in Chapter 6.

5.2. Description

Hubballi and Goyal (Section 4.1.1, [15]) use similarity in interarrival times of flows to detect periodicity by calculating the standard deviation of flow interarrival times for pairs of IP addresses. They monitor the flows to and from a desktop PC both from artificially generated periodic traffic and from real-life traffic of periodic and non-periodic applications.

In our Experiment, we use the one-hour collection of TCP flows in set 42 from the University of Twente data set (discussed in Chapter 3) and wish to find all periodic connections. To implement the technique, we have to group the flow start times per connection, order them chronologically and calculate the sample standard deviation of the interarrival times.

Because a low standard deviation of interarrival times indicates that all the interarrival times are close to one another, we use this to define and detect periodicity with period 1.

The interarrival times of the flows are given by $d_i = t_{i+1} - t_i$, $i = 1, \ldots, n' = n - 1$, with average $\overline{d} = \frac{1}{n'} \sum_{i=1}^{n'} d_i$. The standard deviation periodicity metric s of the connection from A to B in time window W is given by the sample standard

deviation,

$$s = \sqrt{\frac{1}{n' - 1} \sum_{i=1}^{n'} (d_i - \overline{d})^2}. \tag{5.1}$$

A low standard deviation indicates that a connection is periodic. Because the sample standard deviation is only a meaningful estimator when there are enough observations, we require a minimum number of flows N. Further in this chapter, we will see that it can be beneficial to only consider connections that have traffic lasting for longer than a specified minimum time $t_{\min}$. This leads to the following definition of periodicity, used in this chapter. (For the definition of a connection, we refer to Definition 3.1.)

DEFINITION 5.1 (Periodic). *Given N, $t_{\min}$ and T, a connection from host A to B is periodic with period 1 in a time window W if in that time window:*

1. *The number of flows n is at least N: $n \geq N$;*
2. *The flows span at least a time span $t_{\min}$: $t_n - t_1 \geq t_{\min}$;*
3. *The standard deviation periodicity metric s lies below threshold T: $s < T$.*

Hubballi and Goyal suggest only taking into account connections with at least $N = 10$ flows; we do the same. Hubballi and Goyal do not use a minimum time span. At first, neither do we — i.e., we set $t_{\min} = 0$. (We will introduce a minimum time span in Section 5.7.) We consider both the thresholds of $T = 10$ and $T = 3$ seconds listed in the article by Hubballi and Goyal and analyse how the standard deviation periodicity metric is distributed among the connections in the data set. The first results from these findings will lead us to set a new threshold. The time window used is the whole time window of set 42 from the University of Twente data set, which is one hour long.

5.3. Implementation

The data set contains the netflows of all traffic in and out of a university campus network for about one hour. Because we wish to calculate the standard deviation periodicity metric per connection, and our input consist of raw netflow data, the flows need to be sorted on source and destination IP address. To correctly calculate the interarrival times, the flows also need to be sorted on starting time. The complexity of sorting is $\mathcal{O}(n \log n)$. Calculating the sample standard deviations and checking the conditions for periodicity done for all connections can be done with simple loops in $\mathcal{O}(n)$.

The implementation was written in Python, using the pandas library for reading and sorting the data and the SciPy library for calculating the sample standard deviation.

In contrast with Hubballi and Goyal, a data summarization technique was not applied because the input consist of netflow data, which is already a summary of the actual traffic.

5.4. Preliminary results

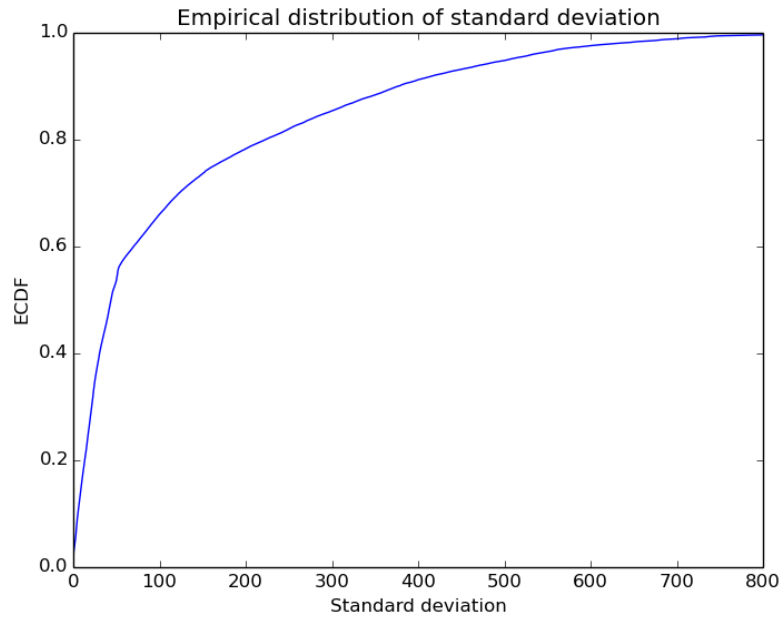

FIGURE 5.1. Empirical cumulative distribution function of the standard deviation periodicity metric of the interarrival times between flows for all host-pairs. On the horizontal axis: the sample standard deviation of flow interarrival times. On the vertical axis: the fraction of host-pairs that has at most the associated sample standard deviation.

Our data set contains 109,585 host-pairs with at least ten flows. Of those, 17,585 (16 percent) are classified as periodic when the threshold is set at ten seconds. With the lower threshold of three seconds, 6,592 pairs (6 percent) are classified as periodic. The full ECDF can be seen in Figures 5.1 and 5.2.

To assess the reliability of the method, we have to consider both false positives and false negatives. Because we are still unsure what aberrations we would like to tolerate and what the data looks like, we made graphic representations of a large number of connections and tried to find patterns which seemed 'periodic' or interesting. We then compare these intuitions to the results of our algorithmic method.

The graphic representation shows the flows of a connection as lines in a consecutive order, with the length of a line indicating the duration of the flow. Statistics on the number of flows and the mean, sample standard deviation and coefficient of variation of interarrival times are also shown on the plot. Using these plots to look at the data, we identified connections that are identified by our method and

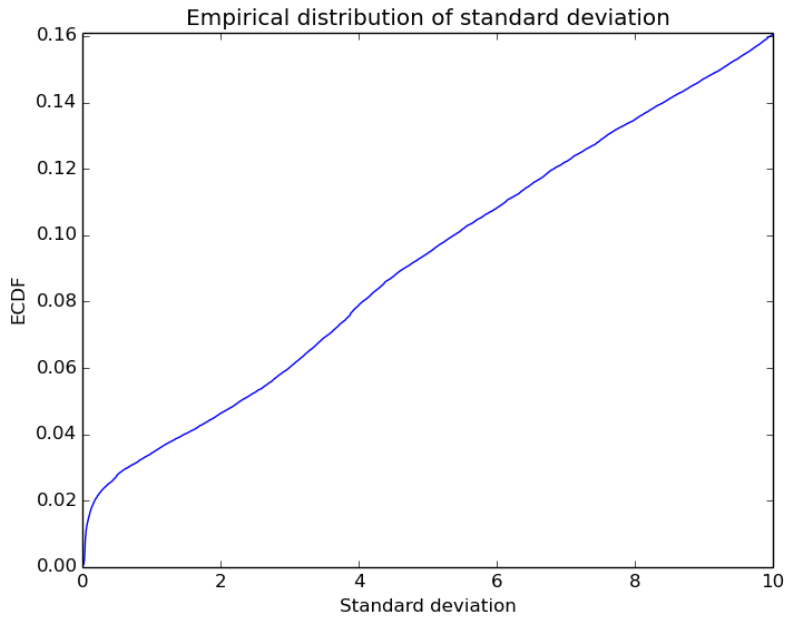

FIGURE 5.2. Zoomed in version of Figure 5.1 for the domain $[0, 10]$ of host-pairs which are labelled as being periodic.

indeed are very regular, discussed in Section 5.4.1. Furthermore, there are some connections that seem very regular when we consider the uneven and even numbered flows separately, but which are undetected by our current method. We would like these to be detected as being 'periodic with period 2'. These are discussed in Section 5.4.2. Other connections are regular in parts of the time window, but show deviations. These are discussed in Section 5.4.3. Finally, some groups were detected but don't seem to be very relevant. These are discussed in Section 5.4.4.

5.4.1. True positives. Connections like that of Figure 5.3 are highly regular. This particular connection has flows with close to exactly 120 seconds in between each of the flow start times. Looking into the flow records associated with these IP addresses, we find that each flow has a source port of 6667 and a varying destination port, indicating traffic with an internet relay chat (IRC) server. All flows consist of one packet of 88 bytes. We consider connections like these to be true positives.

5.4.2. Regular with 'period 2'. The flows in Figure 5.4 seem to have highly regular interarrival times if we consider the uneven and even numbered flows separately. All flows have destination port 6969, normally used for internet relay chat (IRC). The number of packets is 7 for all flows and the number of bytes is 601. Because the flow interarrival times alternate between close to 310 and close to 0

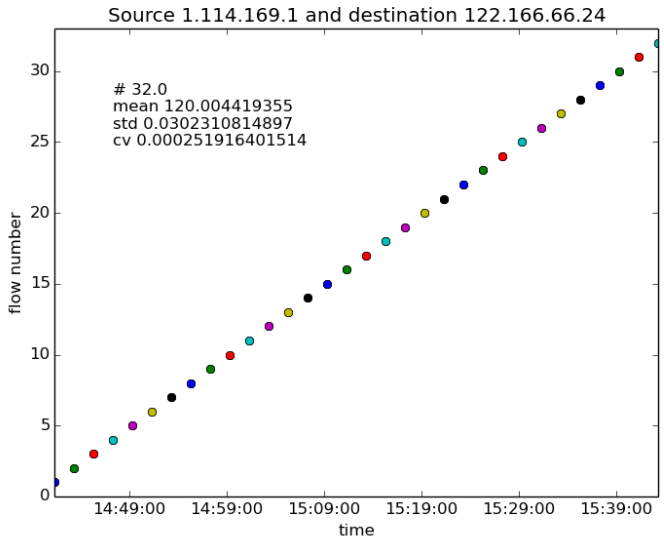

FIGURE 5.3. These flows, most likely from an internet relay chat server, show a strikingly regular pattern. This results in a very low sample standard deviation of the interarrival times between flows, so that the connection is correctly detected as periodic.

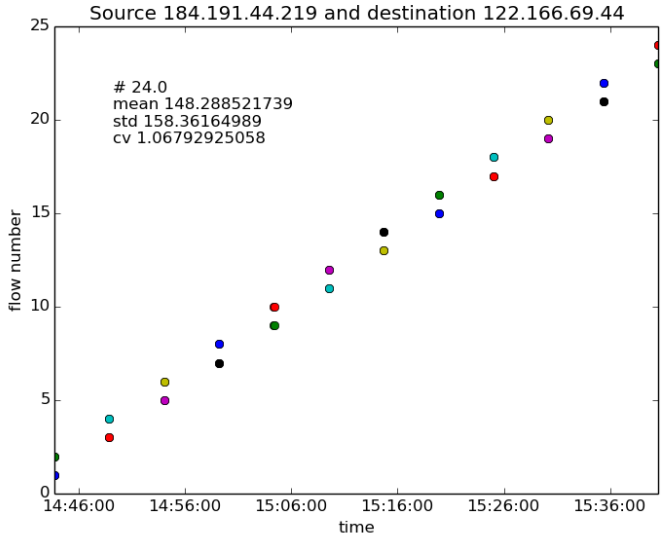

FIGURE 5.4. This sequence of POP3 traffic flows seems very regular for every set of two sequential flows. Because the interarrival times alternate between being short and longer, the sample standard deviation is high, so that the sequence is undetected.

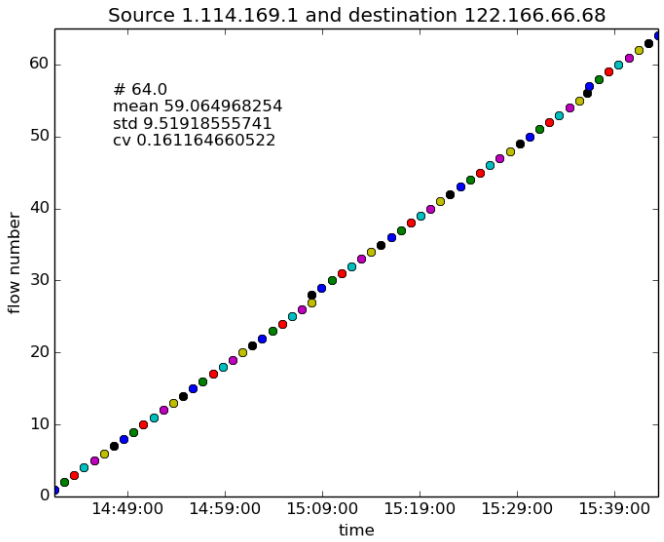

FIGURE 5.5. The sequence of IRC traffic flows depicted here seems very periodic apart from the two 'bumps' around 15:08 and 15:35 o'clock. These cause the sample standard deviation to be relatively high, but still below the threshold of 10 seconds.

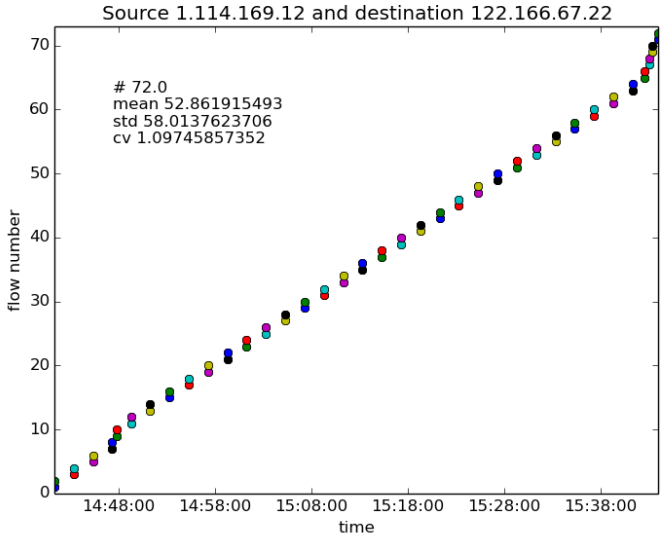

FIGURE 5.6. This sequence of IRC traffic flows seems very regular modulo two in the middle part of the figure, but has 'bumps' and changes near the beginning and end. The sequence is not detected.

seconds, a mean of around 148 with a large sample standard deviation is calculated, and the connection is not classified as being periodic by our method. When we manually split the flows into two groups, those of the uneven numbered and even numbered flows, both groups have a mean of 310 seconds and sample standard deviation of 0.07 seconds. This sequence is thus very regular in interarrival times modulo 2, but is left undetected. Our current method does not address such regularity modulo 2, but we would like it to be detected — meaning that there is room for improvement. Thus we classify this connection as a false negative.

5.4.3. Periodic with deviations. The pattern in Figure 5.5 shows two deviations, slight 'bumps' around 15:08 and 15:35 o'clock, but seems to be very regular when those bumps are ignored. The sample standard deviation is close to 10 — above the lower threshold of 3, but below the higher threshold of 10. The flows originate from port 6667, likely indicating an IRC server. All flows consist of 1 packet of 99 bytes, except for the two flows causing the 'bumps'. When these two flows are ignored, the mean interarrival time is 61 seconds with a sample standard deviation periodicity metric of 0.49 seconds. The standard deviation periodicity metric is similarly low when a different time window is chosen that does not include those two deviations. This indicates the method is susceptible to noise and the choice of time window.

The flows in Figure 5.6 are from IRC traffic on port 6667. The flows mostly consist of 4 packets with a total size of 307 or 315 bytes, but show no clear period of 2 in those dimensions. As was the case for the POP3 flows in Figure 5.4, the sample standard deviation is high because short interarrival times are alternated by longer interarrival times. Unlike those flows, splitting the flows into groups of uneven and even flows yields sample standard deviations of 30.5 seconds, still significantly higher than the threshold of 10 seconds. When looking at the figure, we see that a bump occurs around 14:48 and that the flows after 15:40 o'clock have lower interarrival times. Adjusting the time window such that the first 10 and last 8 flows aren't taken into account, we find a mean of 58.9 and sample standard deviation of 60.6 seconds. After splitting the uneven and even flows, we find a mean of 120.0 and sample standard deviation of 0.063 for both groups. Thus this sequence shows strong regularity modulo 2 in the interarrival times in a subset of our time window, but it is not detected because our method only considers period 1. Even when this is accounted for by looking at even and uneven flows separately, the deviating flows at the beginning and end of the sequence greatly influence the sample standard deviation, pushing it over the threshold. This is another example that the method is susceptible to noise and the choice of time window.

5.4.4. Quick succession false positives. The flows shown in Figures 5.7 and 5.8 follow on each other with such a small time in-between that the variance of inter-arrival times is inevitably small. Relatively to the mean of the interarrival times, the sample standard deviation is quite high, as is seen by the high values

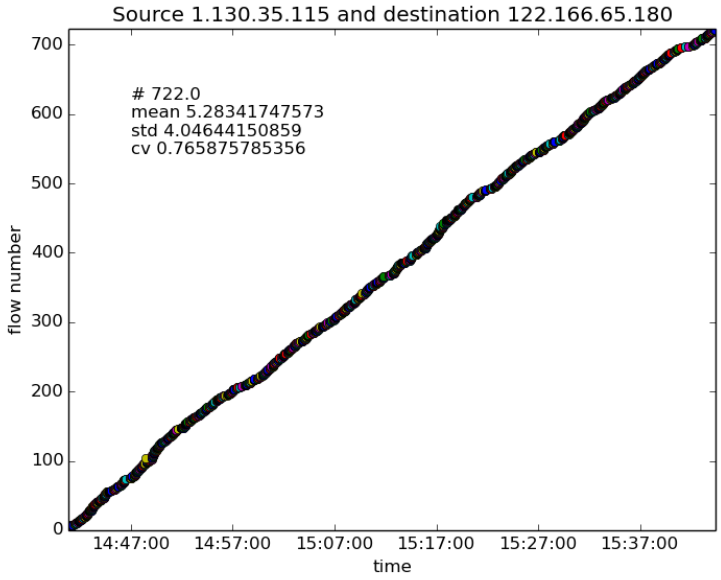

FIGURE 5.7. The flows in this sequence, likely IRC traffic, rapidly succeed each other, causing a low sample standard deviation. Thus, the sequence is marked as periodic, though it's not very regular.

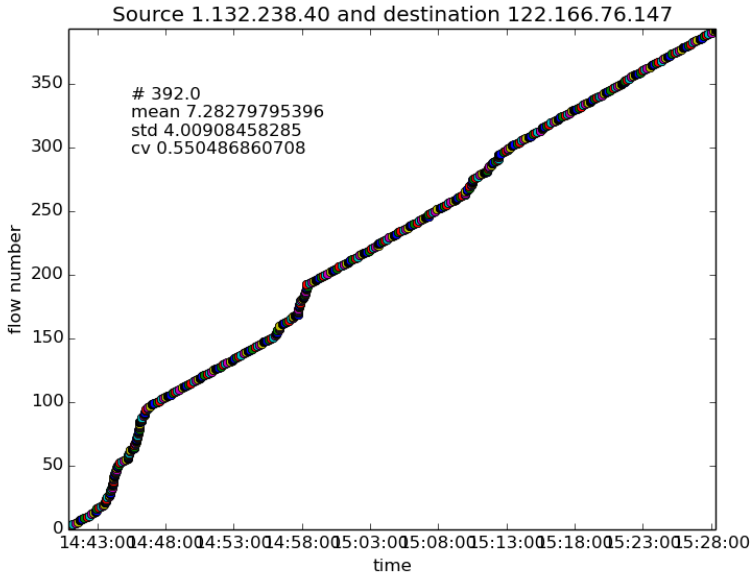

FIGURE 5.8. The flows in this sequence, likely HTTP traffic, rapidly succeed each other, causing a low sample standard deviation. Thus, the sequence is marked as periodic, though it's not very regular.

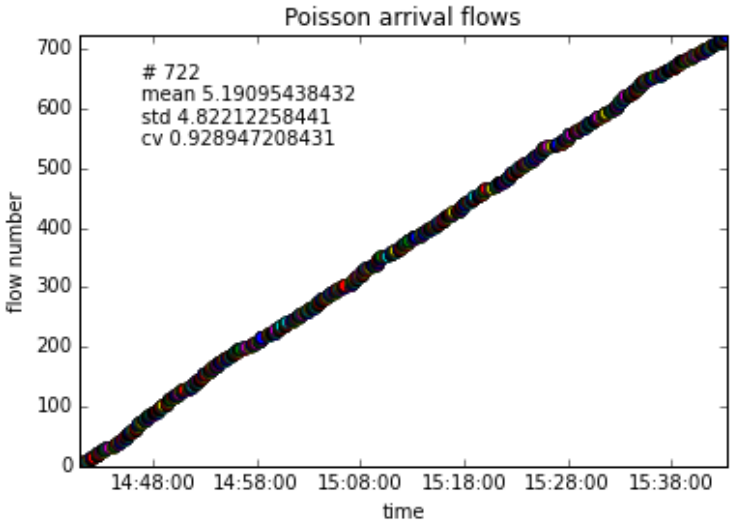

FIGURE 5.9. Generated Poisson arrivals with the same mean interarrival time as that of the flows in Figure 5.7.

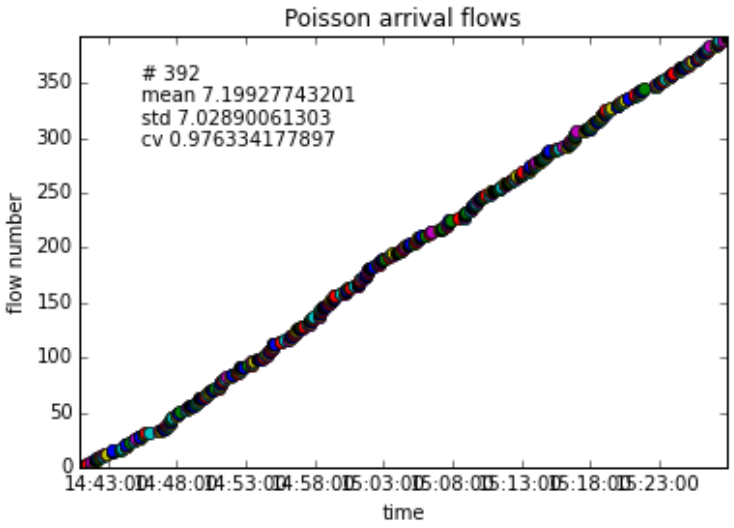

FIGURE 5.10. Generated Poisson arrivals with the same mean interarrival time as that of the flows in Figure 5.8.

of the coefficient of variation. The number of packets and bytes per flow vary, further indicating that these flows are not very regular and must be considered false positives. The traffic on Figure 5.7 has source port 6666, indicating it is likely to be IRC traffic; the traffic on Figure 5.8 has source port 80, likely indicating HTTP, the hypertext transfer protocol (a web server).

To test the hypothesis that these connections are false positives and only detected because of the quick succession of flows, we compare them to random data: the same number of arrivals from a Poisson process with the same mean. As seen in Figures 5.9 and 5.10, these flows with non-deterministic interarrival times have only slightly higher sample standard deviation and coefficient of variation and would also be classified as periodic under the threshold of 10. Thus we can safely say that the detected 'periodicity' in the traffic from these two examples are false positives.

We would like to see if we can generalise our findings on false positives because of quick successions, by analysing the behaviour of non-regular data from a Poisson process. The interarrival times of a Poisson process are exponentially distributed. An exponential distribution with parameter λ has mean λ^{-1} and standard deviation λ^{-1}. Thus, we would expect (or at least hope) for our standard deviation periodicity metric s to be close to λ^{-1} for such processes.

The sample standard variance s^2 gives an unbiased estimator of the sample variance, i.e., if $d_1, \ldots, d_{n'}$ are i.i.d. with standard deviation σ, $\mathbb{E}(s^2) = \sigma^2$. However, s is not an unbiased estimator of the standard variance, i.e., $\mathbb{E}(s) \neq \sigma$ in general. Unbiased estimators of the standard deviation exist when the distribution of the d_i is known, but in our case this distribution is not known. Furthermore, we do not know whether the interarrival times are independent.

In the case of a Poisson process, the interarrival times are exponentially distributed with standard deviation λ^{-1}. Thus if $\lambda^{-1} >> T$, we would expect the chance of s being less than T to be low; especially if there are many flows (a large n), so that s is a better approximation of λ^{-1}. The probability associated with this event, $P(s < T)$ for a Poisson process, is

$$\begin{aligned} P\left(s < T\right) &= P\left(s^2 < T^2\right) \\ &= P\left(\frac{1}{n'-1}\sum_{i=1}^{n'}\left(X_i - \overline{X}\right)^2 < T^2\right), \end{aligned}$$

where the X_i are i.i.d. exponentials with parameter λ. We simulated this probability using Monte Carlo simulation for different thresholds $T = 0.1$, $T = 5$ and $T = 10$ and different numbers of interarrival times $n' = 9$, $n' = 99$ and $n' = 999$ (corresponding to the number of interarrival times for connections with 10, 100 and 1000 flows). The result can be seen in Figure 5.11. Because the results for $T = 0.1$ are not very visible on this figure, they are separately shown in Figure 5.12.

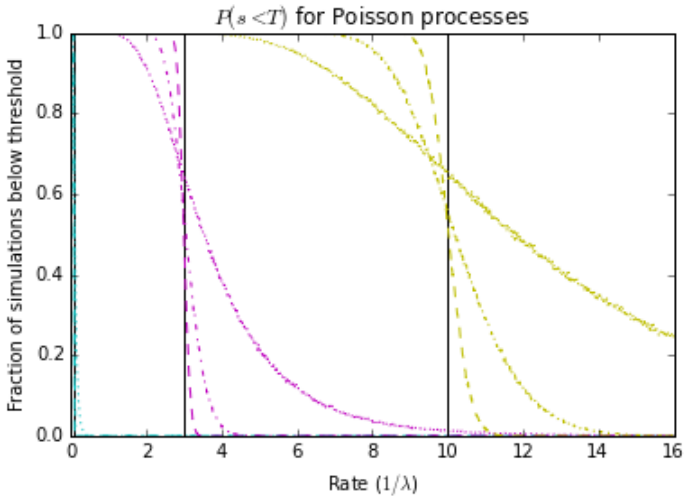

FIGURE 5.11. Monte Carlo simulation of $P(s < T)$ for Poisson processes with different rates. Different thresholds are indicated with colours cyan for $T = 0.1$, pink for $T = 3$ and yellow for $T = 10$. Different number of observations are indicated with a dotted line for $n = 10$, dot-dashed line for $n = 100$ and dashed line for $n' = 1000$.

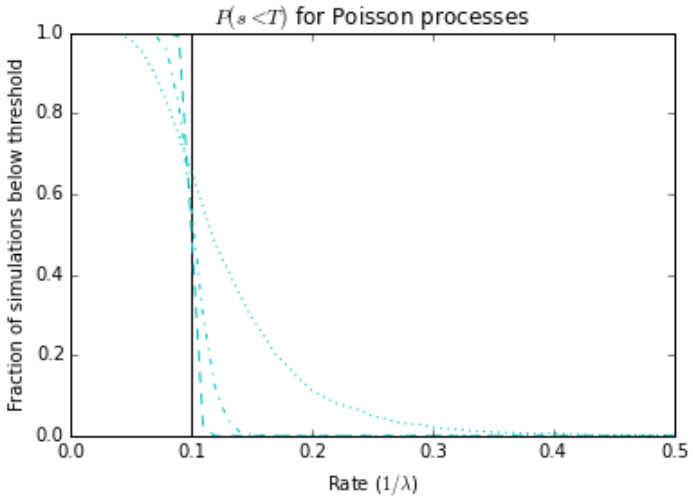

FIGURE 5.12. Monte Carlo simulation of $P(s < T)$ for Poisson processes with different rates, where $T = 0.1$. The dotted line indicates $n = 10$ observations, the dot-dashed line $n = 100$ observations and the dashed line $n = 1000$ observations.

These results indicate several options to counter false positives from a quick succession of flows. Note that for Poisson processes, $P(s < T)$ is only significant if the threshold T is close to the rate of the Poisson process, which is also the mean interarrival time of the process. Thus, low scores for Poisson processes can only come from processes with very rapid successions, which is an abnormal situation. By choosing the threshold T low enough, we may be able to filter out most or all of these processes. Another option is to set requirements of the mean interarrival time relative to the sample standard deviation s (i.e. on the coefficient of variation). For Poisson processes, we have shown that the mean is close to s, so we could require the coefficient of variation to be at most 0.5 (i.e., s below two times the mean.)

5.5. Choosing a lower threshold

We have seen that it takes only a small disturbance of periodicity to greatly increase the standard deviation periodicity score. We have also seen that when there are no such disturbances, the score is much lower than the threshold of 3 and 10. This means that to detect periodic traffic without disturbances, we could choose a lower threshold. The periodic traffic with disturbances, which is currently detected, is detected more because of luck than good strategy, because an outlier easily pushes the score above the threshold, but the high threshold also introduces false positives. With a lower threshold, such traffic with a disturbance can still be detected in the time window where it is actually very regular; that is, in a time window without the disturbance. Thus we wonder: what is a reasonable lower threshold for periodic traffic?

The thresholds of 10 and 3 were based on empirical observations by Hubballi and Goyal. We now wonder what a good threshold would be based on the possible causes of variation in interarrival times. The timestamps in the netflow data are given in milliseconds. Assume that a computer program is programmed to poll a server exactly once every so often. In this case, there is no variation in the interarrival times due to the program itself. There could be a very small delay in the execution of the program, because the machine it runs on is under heavy load and the program has to share the computer's resources with other programs running on the machine. In sending the message over the network, there will be a delay due to the limited speed of light and due to processing in the end-points and the routers in-between. The variation in this end-to end delay is called the packet delay variation. There can also be a delay in the registering of the arrival time by the netflow monitor. This accuracy in registration time is in the order of 100 microseconds [31]. We could not find general estimates of packet delay variation, but based on [39, 40] it is likely to be well below 100 milliseconds. The delay in the computer is likely to be much lower, because CPUs process billions of instructions per second and give precedence to short processes, such as polling behaviour.

Based on these observations, we conclude that the variation of interarrival times for a periodic process is in the order of tens of milliseconds. Combined with the empirical observation from the graphical representations that periodic traffic without disturbances has scores below 0.1 seconds, we choose a threshold of 0.1 seconds.

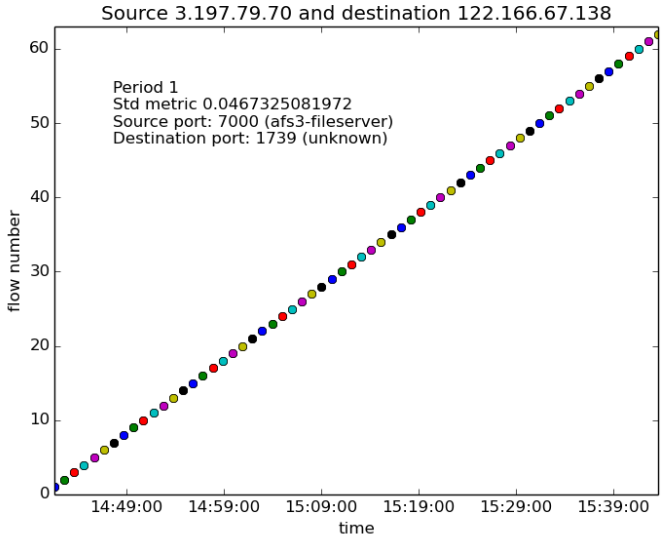

FIGURE 5.13. After lowering the threshold to 0.1 seconds, almost all connections determined to be periodic look like this one: a long-term succession of flows with near-constant interarrival times.

5.6. Results with lower $T = 0.1$ second threshold

With the lower threshold of $T = 0.1$ second, the numbers of host-pairs detected as periodic goes down to 1,718 or 1.6%, compared to 16% and 6% for the ten second and three second thresholds. The connections of quick succession from Figures 5.7 and 5.8 are no longer detected, nor is the traffic from Figure 5.5, which has disturbances of its periodicity.

By examining a large sample of plots of the connections now labelled as periodic, we found that most are from traffic over a sustained time with a highly regular succession of flows, such as in Figure 5.13 — exactly the kind of polling behaviour we wish to find.

However, some connections are still detected which seem to be false positives, like the flows depicted in Figure 5.14. These kind of connections are very short-lived; for most of these connections, all flows occur in an interval of only a few seconds. Because all flows start in such a small time window, the interarrival times are by necessity all very small and thus have a low standard deviation.

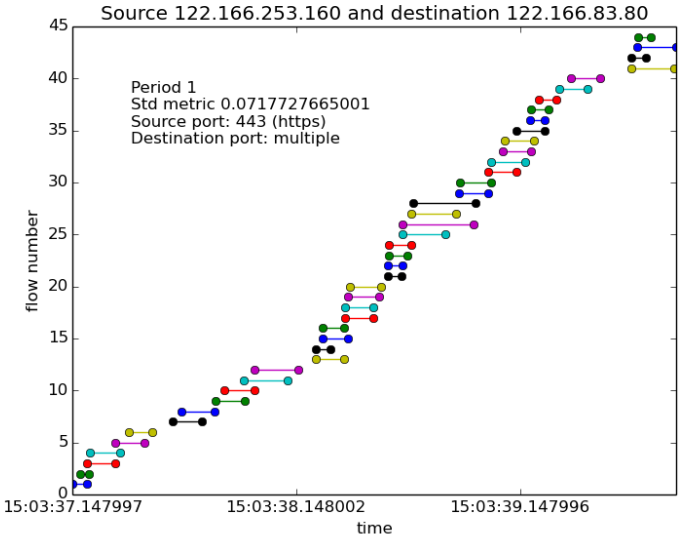

FIGURE 5.14. The flows in this figure all start in an interval of two seconds. Because of the very small time in-between each flow, the standard deviation is necessarily small.

The reason these false positives occur is because they have a low sample standard deviation but also a very small mean interarrival time. One way to solve this could be to analyse not the sample standard deviation but the sample coefficient of variation. However, this may introduce other problems and it is not clear what would be a good way of estimating a value for this parameter. Luckily, we note that the false positives only occur when many flows occur in a very short interval (a few seconds or less). Since the polling behaviour we are interested in lasts for much longer than the duration of the connections described in the previous section, we simply remove all connections for which the total duration (time from first to last flow) is below a minute, by setting $t_{\min}$ to 60 seconds.

5.7. Further refinements — final implementation

In our final implementation, we define a connection to be periodic if there are at least $N = 10$ flows spanning a time of at least $t_{\min} = 60$ seconds with a standard deviation periodicity metric below $T = 0.1$ seconds.

5.8. Final results

By excluding traffic which doesn't last for at least 60 seconds, the number of periodic connections drops from 1,718 out of 109,585 to 1,496 out of 104,924 or 1.4%. The remaining detected connections are all from long-lived traffic with highly regular interarrival times, like that of Figure 5.13.

5.9. Conclusion

We have successfully applied Hubballi and Goyal's standard deviation technique for detecting periodicity in the traffic of one host to a data set containing the traffic of a whole university network. From empirical analysis of false positives, combined with deductions on the expected variation in interarrival times, we propose a new, much lower threshold of 0.1 seconds. Combined with a new setting for removing traffic that appears only in a small time interval, this removes nearly all false positives.

In our university campus network, 1.4% of connections are periodic under the improved threshold and settings, indicating that a significant amount of the connections is from machine-generated polling behaviour.

An analysis of the traffic in the data set brought to mind connections that are very regular when analysed modulo 2. The existence of such connections warrants further enquiry in finding such connections.

We noted some connections that are not periodic under our chosen time window because of irregularities, but are periodic in subsets of this time window. We stress that the method detects periodic behaviour in a given time window, but that connections may be periodic in some time windows and not in others, so that detection is dependent on the chosen time window.

With the adaptations we made to the standard deviation periodicity metric, it is an effective technique to detect periodicity in network traffic in a specified time window. In the next chapter, we will try to expand the metric to include connections that are regular modulo some period p other than one.

CHAPTER 6

Experiment 2: Extension to larger periods

6.1. Goal

In Experiment 1 (Chapter 5), connections with period 2 were found in the data set, but they were not detected by the detection method. We wish to find out if there are more connections like those and if there are also connections with even larger periods present in the data.

For this reason, we introduce a new method of periodicity detection. This method uses the autocorrelation and the existence of consecutive local peaks (explained below) to find candidate periods for the connections. An adapted version of the standard deviation metric used in Experiment 1 is then used to classify the data as periodic or not.

6.2. Autocorrelation and its computation

For a sequence $x = (x_0, x_1, \ldots, x_{N-1})$ of length N, the (circular) autocorrelation function for lag k, $r_k(x)$, is defined as

$$r_k(x) = \sum_{i=0}^{N-1} x_i \cdot x_{i-k}. \tag{6.1}$$

Here and in the rest of this chapter, all indices are assumed to be modulo N. In other words, the autocorrelation of x for lag k is the inner product

$$\langle x, x^k \rangle = ||x|| \cdot ||x_k|| \cdot \cos\theta$$

of x with x^k, x shifted over k positions backward, where θ is the angle between x and x^k in $\mathbb{R}^n$.

Example 6.1 (Running example). Consider the sequence of inter-arrival times $d = (61, 28, 59, 31, 60, 29, 61, 31, 58, 30)$ of length $N = 10$. With a lag of $k = 3$, the vector turns into $d^3 = (31, 58, 30, 61, 28, 59, 31, 60, 29, 61)$. The inner product of d and d^3 is the autocorrelation for lag 3 of $61 \cdot 31 + 28 \cdot 58 + \cdots + 30 \cdot 61 = 17830$.

Since a shift over N positions returns the original vector, shifts are equal modulo N and we need only study the lags $k = 0, \ldots, N-1$. We refer to the set of autocorrelations for these lags as the autocorrelation function (ACF),

$$\mathrm{ACF}(x) = (r_0(x), r_1(x), \ldots, r_{N-1}(x)). \tag{6.2}$$

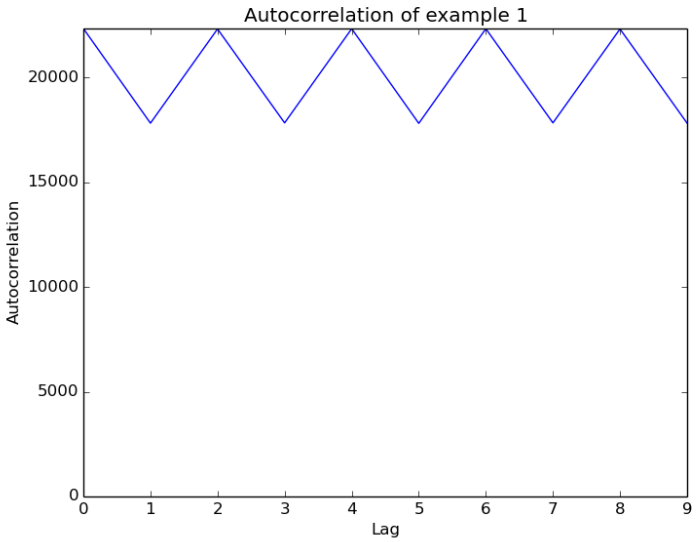

FIGURE 6.1. Graphical representation of the autocorrelation function from Example 6.1

EXAMPLE 6.2 (Running example (ct.)). The autocorrelations for the lags $k = 0, \ldots, 9$ of d are given by

$$\mathrm{ACF}(d) = (22334, 17817, 22311, 17830, 22323, 17808, 22323, 17830, 22311, 17817).$$

The function is plotted in Figure 6.1. There are clear peaks at lag two and multiples of two.

In the example, we see that the ACF is symmetric around $k = 0$. That is, for lag '-1' or $N - 1$ the autocorrelation is the same as for lag 1; lag '-2' or $N - 2$ has the same ACF value as lag 2; lag '-3' or $N - 3$ has the same ACF values as lag 3, and so on. The maximum autocorrelation is achieved for $k = 0$. These are general properties of the ACF, which will be shown below.

6.2.1. Geometric interpretation of autocorrelation. For two elements A and B of $\mathbb{R}^n$, their inner product is given by $\langle A, B\rangle = ||A|| \cdot ||B|| \cos\theta$, where $0 \leq \theta \leq \pi$ is the angle between A and B. For a sequence x and its shifted version by k positions x^k, we thus have $r(k) = \langle x, x^k\rangle = ||x||^2 \cos\theta$, where θ is the angle x and x^k make.

LEMMA 6.1. *The autocorrelation is maximal at lag $k = 0$.*

PROOF. From the geometric interpretation, it follows that $r(k) \leq ||x||^2$, with equality when $\cos\theta = 1$. When $k = 0$, $x^k = x$ is not shifted at all, so that $\theta = 0$. Hence, $r(0) = ||x||^2$, which is maximal. □

Dividing $r(k)$ by $||x||^2$ thus gives a normalised autocorrelation, which is the way some authors define the autocorrelation. Another element of normalisation can be to subtract the mean from all observations, i.e., to replace $x_i \cdot x_{k+i}$ by $(x_i - \overline{x})(x_{k+i} - \overline{x})$ in Equation (6.1).

LEMMA 6.2. *The autocorrelation function is symmetric around $k = 0$. (The statement is to be read modulo N: the autocorrelation for lag $k = 1$ equals that of lag $N - 1$, that of lag 2 equals that of lag $N - 2$, and so on.)*

PROOF. Consider a negative lag of $-k$ and a positive lag of $+k$. The corresponding autocorrelations are $r(k) = \sum_{j=0}^{N-1} x_j x_{j+k}$ and $r(-k) = \sum_{i=0}^{N-1} x_i \cdot x_{i-k}$. For $k \leq i \leq n-1$, $x_i \cdot x_{i-k}$ is equal to $x_j \cdot x_{j+k}$ for $j = i-k$, with $0 \leq j \leq n-k-1$. For $0 \leq i \leq k-1$, $x_i \cdot x_{i-k}$ corresponds to $x_i \cdot x_{n+i-k}$ since $i-k \mod n = n+i-k$ for $0 \leq i \leq k-1$. Similarly, for $n-k \leq j \leq n-1$, $x_j \cdot x_{j+k}$ corresponds to $x_j \cdot x_{j+k-n}$. Letting $j = n-k+i$, it follows that for $0 \leq i \leq k-1$, $x_i \cdot x_{n+i-k}$ is equal to $x_j \cdot x_{j+k-n}$, with $n-k \leq j \leq n-1$. Hence every term of the sum in $r(k)$ corresponds to exactly one term of the sum in $r(-k)$, so that $r(k) = r(-k)$. □

Because the autocorrelation function is symmetric, it is sufficient to calculate lags 0 up to and including $k = \lfloor n/2 \rfloor$.

6.2.2. Autocorrelation and the discrete Fourier transform. Computing the autocorrelation for a lag k has complexity $\mathcal{O}(n)$, so that calculating the autocorrelation for all lags 0 up to and including $\lfloor n/2 \rfloor$ has complexity $\mathcal{O}(n^2)$. For this reason, it is common to compute the autocorrelation by using a discrete Fourier transform [41], for which fast $\mathcal{O}(n \log n)$ algorithms exist. This section serves to prove Theorem 6.1, which provides a way of calculating the autocorrelation using Fourier transforms.

For a sequence of (complex) numbers $x = (x_0, \ldots, x_{N-1})$, the discrete Fourier transform $\mathcal{F}(x) = X$ is given by a sequence of (complex) numbers $X = (X_0, \ldots, X_{N-1})$, given by

$$X_m = \sum_{n=0}^{N-1} x_n \cdot e^{-2\pi i m n/N}. \tag{6.3}$$

LEMMA 6.3. *The Fourier transform is invertible, and the inverse Fourier transform $\mathcal{F}^{-1}(X) = x$ of a sequence of complex numbers $X = (X_0, \ldots, X_{N-1})$ is given by a sequence $x = (x_0, \ldots, x_{N-1})$ defined as*

$$x_n = \frac{1}{N} \sum_{m=0}^{N-1} X_m \cdot e^{2\pi i m n/N}. \tag{6.4}$$

PROOF. Let x be a sequence of complex numbers of length N. Then its Fourier transform $\mathcal{F}(x)$ is a sequence of complex numbers of length N and we wish to show

that $\mathcal{F}^{-1}\mathcal{F}(x) = x$. If we let $X_m = \mathcal{F}(x)(m)$, then

$$\begin{aligned}(\mathcal{F}^{-1}\mathcal{F}(x))(n) &= \frac{1}{N}\sum_{m=0}^{N-1} X_m \cdot e^{2\pi imn/N} \\ &= \frac{1}{N}\sum_{m=0}^{N-1}\sum_{k=0}^{N-1} x_k \cdot e^{-2\pi imk/N} \cdot e^{2\pi imn/N} \\ &= \frac{1}{N}\sum_{m=0}^{N-1}\sum_{k=0}^{N-1} x_k \cdot e^{2\pi im(n-k)/N} \\ &= \frac{1}{N}\sum_{k=0}^{N-1} x_k \sum_{m=0}^{N-1} e^{2\pi im(n-k)/N}.\end{aligned}$$

The exponential equals 1 if $n = k$. If $n \neq k$, we can write the inner sum as a geometric series, using $a = e^{2\pi i(n-k)/N}$ for ease of notation.

$$\begin{aligned}\sum_{m=0}^{N-1} e^{2\pi im(n-k)/N} &= \sum_{m=0}^{N-1} (e^{2\pi i(n-k)/N})^m \\ &= \sum_{m=0}^{N-1} a^m \\ &= \frac{1-a^N}{1-a} \\ &= \frac{1-e^{i2\pi(n-k)}}{1-e^{i2\pi(n-k)/N}}.\end{aligned}$$

Since n and k are integers between 0 and $N-1$ and $n \neq k$, $n-k$ is integer and $n-k \neq 0 \pmod N$, so that $e^{i2\pi(n-k)} = 1$ and $e^{i2\pi(n-k)/N} \neq 1$ — the numerator equals 0 and the denominator does not. Hence, the fraction is equal to zero in this case. Therefore, the inner sum equals 0 if $n \neq k$ and is a sum of ones if $n = k$. In conclusion,

$$\begin{aligned}(\mathcal{F}^{-1}\mathcal{F}(x))(n) &= \frac{1}{N}\sum_{k=0}^{N-1} x_k \sum_{m=0}^{N-1} \mathbb{1}_{n=k} \\ &= \frac{1}{N}\sum_{k=0}^{N-1} x_k \cdot N \cdot \mathbb{1}_{n=k} \\ &= x_n.\end{aligned}$$

□

DEFINITION 6.1. *The convolution $z \star w$ of two sequences z and w of length N is defined by*

$$(z \star w)(m) = \sum_{n=0}^{N-1} z(m-n)w(n), \qquad m \in \mathbb{Z}. \tag{6.5}$$

(As elsewhere in this chapter, the indices are to be taken modulo N.)

LEMMA 6.4. *The Fourier transform of a convolution equals the element-wise product of the Fourier transforms of the convoluted terms; i.e., for sequences z and w of length N,*

$$\mathcal{F}(z \star w) = \mathcal{F}(z) \cdot \mathcal{F}(w). \tag{6.6}$$

PROOF. The proof is chiefly an exercise in rewriting.

$$\begin{aligned}
(\mathcal{F}(z \star w))(m) &= \sum_{n=0}^{N-1} (z \star w)(n) e^{-2\pi imn/N} \\
&= \sum_{n=0}^{N-1} \sum_{k=0}^{N-1} z(n-k)w(k)e^{-2\pi imn/N} \\
&= \sum_{n=0}^{N-1} \sum_{k=0}^{N-1} z(n-k)w(k)e^{-2\pi im(n-k)/N} e^{-2\pi imk/N} \\
&= \sum_{k=0}^{N-1} \sum_{n=0}^{N-1} z(n-k)e^{-2\pi im(n-k)/N} w(k)e^{-2\pi imk/N} \\
&= \sum_{k=0}^{N-1} \sum_{l=-k}^{N-1-k} z(l)e^{-2\pi iml/N} w(k)e^{-2\pi imk/N}
\end{aligned}$$

Note that the first exponential function is periodic in l with period N, so that we can take both the l in $z(l)$ and in the exponential function to mean l modulo N. Thus, the negative values of $l = -1, -2, \ldots, -k$ correspond to $l = N-1, N-2, \ldots, N-k$. Hence,

$$\begin{aligned}
(\mathcal{F}(z \star w))(m) &= \sum_{k=0}^{N-1} \sum_{l=0}^{N-1} z(l)e^{-2\pi iml/N} w(k)e^{-2\pi imk/N} \\
&= \sum_{k=0}^{N-1} \mathcal{F}(z)(m)w(k)e^{-2\pi imk/N} \\
&= \mathcal{F}(z)(m) \cdot \sum_{k=0}^{N-1} w(k)e^{-2\pi imk/N} \\
&= \mathcal{F}(z)(m) \cdot \mathcal{F}(w)(m).
\end{aligned}$$

□

LEMMA 6.5. *For a sequence of real numbers z, the complex conjugate of its Fourier transform is equal to the sequence in reverse order, i.e. if we define w by $w(n) = z(-n)$, then $\mathcal{F}(w) = \overline{\mathcal{F}(z)}$.*

PROOF. Since z is real, $\overline{z(n)} = z(n)$. Furthermore, $\overline{e^{-ix}} = e^{ix}$ for $x \in \mathbb{R}$, so that

$$\begin{aligned}\mathcal{F}(w)(m) &= \mathcal{F}(z)(-m) \\ &= \sum_{n=0}^{N-1} z(n) e^{-2\pi i(-m)n/N} \\ &= \overline{\sum_{n=0}^{N-1} z(n) e^{-2\pi imn/N}} \\ &= \overline{\mathcal{F}(z)}(m).\end{aligned}$$

□

THEOREM 6.1. *The ACF can be computed using Fourier transforms in the following way, where $\cdot$ is element-wise multiplication,* $\mathrm{ACF}(z) = \mathcal{F}^{-1}(\mathcal{F}(z) \cdot \overline{\mathcal{F}(z)})$.

PROOF. By definition,

$$r(m) = \sum_{n=0}^{N-1} z(n) z(n+m).$$

Note that the indices of z are taken modulo N, so that $z(-1) = z(N-1)$, $z(-2) = z(N-2)$, etc. Hence for $n = 1, 2, \ldots, N-1$, $z(n)z(n+m)$ equals $z(-n)z(-n+m)$ for $n = N-1, N-2, \ldots, 1$. Thus,

$$r(m) = \sum_{n=0}^{N-1} z(m-n) z(-n).$$

For convenience, we define w by $w(n) = z(-n)$.

$$\begin{aligned}r(m) &= \sum_{n=0}^{N-1} z(m-n) w(n) \\ &= (z \star w)(m).\end{aligned}$$

By Lemma 6.4 above,

$$\mathcal{F}(z \star w) = \mathcal{F}(z) \cdot \mathcal{F}(w).$$

By Lemma 6.5 above, $\mathcal{F}(w) = \overline{\mathcal{F}(z)}$ since z is real, and thus

$$F(z \star w) = \mathcal{F}(z) \cdot \overline{\mathcal{F}(z)}.$$

Taking inverse Fourier transforms on both sides leaves us to conclude that

$$\mathrm{ACF}(z) = \mathcal{F}^{-1}(\mathcal{F}(z) \cdot \overline{\mathcal{F}(z)}).$$

□

Though calculating $\mathcal{F}(x)_k = X_k$ for any k requires $\mathcal{O}(n)$ calculations, calculating $\mathcal{F}(x)$ can be done computationally more efficient than the $\mathcal{O}(n^2)$ which would be required for computing all values of X_k separately. Several algorithms, called fast Fourier transforms, exist for computing $\mathcal{F}(x)$ with complexity $\mathcal{O}(n \log n)$. Theorem 6.1 thus enables efficient computation of the ACF. Exploiting the symmetry of the ACF (Lemma 6.2) further reduces computational efforts.

6.2.3. Consecutive local peaks. We wish to define a connection to be periodic if its interarrival times are regular when considered modulo some period p. That is, if any interarrival time d_i is almost the same as d_{i+p}, d_{i+2p}, d_{i+3p}, and so on, we say that the connection is periodic with period p. Necessarily, this means that d_i is also almost the same as d_{i+2p}, d_{i+4p}, d_{i+6p}, and so on. Thus we expect such a connection to have a relatively high autocorrelation not only for lag p, but also for lag $2p$, lag $3p$, and so on (cf. [33], Section 4.2.1).

Building on this idea, we define consecutive local peaks to find candidate periods.

Definition 6.2. *A series x has consecutive local peaks for lag k if its autocorrelation function has local peaks at positions k, $2k$ and $3k$. There is a local peak at position k if $r(k-1) \leq r(k)$ and $r(k) \geq r(k+1)$.*

(Note that there can not be a consecutive local peak for a lag of 1 unless $r(0) = r(1)$. By Lemma 6.1, this implies that x shifted by one position equals itself, which only happens in the special case where all entries of x are equal.)

6.3. Validating the candidate period

Consecutive local peaks indicate a candidate period, but verification is still needed. It is possible to set a threshold on the value of the ACF for the consecutive local peaks, but this value can vary greatly depending on the length of the sequence of interarrival times. If the distribution of interarrival times for non-periodic traffic were known, this could be used to do a hypothesis test. However, this distribution is not known. Another possibility is to randomly scramble the interarrival times and calculate the corresponding ACF's for the randomly scrambled sequences, comparing them to the actual value found. E.g. if the interarrival times are randomly scrambled and the corresponding autocorrelation is computed a 100 times, a score above the second-largest autocorrelation found indicates an empirically 1% chance that this is due to chance. Unfortunately, this last method is computationally expensive.

Since we already used the standard deviation to classify connections as periodic, we propose to extend this approach to periods other than one. For a candidate period of k, this is done by splitting the interarrival times according to their position modulo k. E.g., the first group consists of interarrival times 0, k, $2k$, and so on; the second group consists of interarrival times 1, $k+1$, $2k+1$, and so on. As in the standard deviation periodicity metric in Equation (5.1), the metric is based on the sample standard deviation.

DEFINITION 6.3. *For a period p and flows with interarrival times $d_1, d_2, \ldots, d_{n'}$, let m_i denote the amount of flows that have index $i \pmod p$ and $\overline{d_i}$ the mean of interarrival times with index $i \pmod p$. The standard deviation periodicity metric for period p is then given by the sum of the individual sample variances,*

$$s = \sqrt{\sum_{i=1}^{p} \frac{1}{m_i - 1} \sum_{j=0}^{m_i - 1} (d_{i+jp} - \overline{d_i})^2}. \tag{6.7}$$

6.4. Method

As in Experiment 1, only connections which last at least one minute and contain at least 10 flows are considered. The connections that were not identified as periodic with period 1 in Experiment 1 with the threshold of 0.1 seconds are then analysed. This is done by calculating their autocorrelation function using Fourier transforms and determining the lowest consecutive local peak. If there is none, the connection is not periodic. If the lowest periodic peak is found to be k, the standard deviation periodicity metric is calculated for the connection, with parameter k. If the resulting score is below the threshold of 0.1 seconds, the connection is classified as being periodic with period k.

6.5. Implementation

The same data set as in Experiment 1 was used. Calculating the autocorrelation function using fast Fourier transforms has complexity $\mathcal{O}(n \log n)$. The other operations — determining the lowest consecutive local peak and calculating the periodicity metric have complexity $\mathcal{O}(n)$, giving a total complexity of $\mathcal{O}(n \log n)$. The implementation was done in Python, using the numpy and pandas libraries for the Fourier transformations and data processing.

6.6. Results

Of the 103,428 connections in the data set to which the successive local peak detection was applied, 45,505 were identified to have successive local peaks. After the standard periodicity metric was applied to them, 462 were found to be periodic.

In Experiment 1, we noted that there are connections in the data set with periodicity 2 in the data set and that these were not detected. Indeed, the method

in this chapter finds 428 periodic connections with period 2. Furthermore, connections with even greater periods are found; 22 of period 3, 9 of period 4, 2 of period 5 and 1 of period 13. A histogram can be found in Figure 6.2. Of each of the visually different patterns found in the data set, a representative is plotted in Figures 6.3 to 6.14. All of the discovered connections are very regular; no seemingly false positives were found.

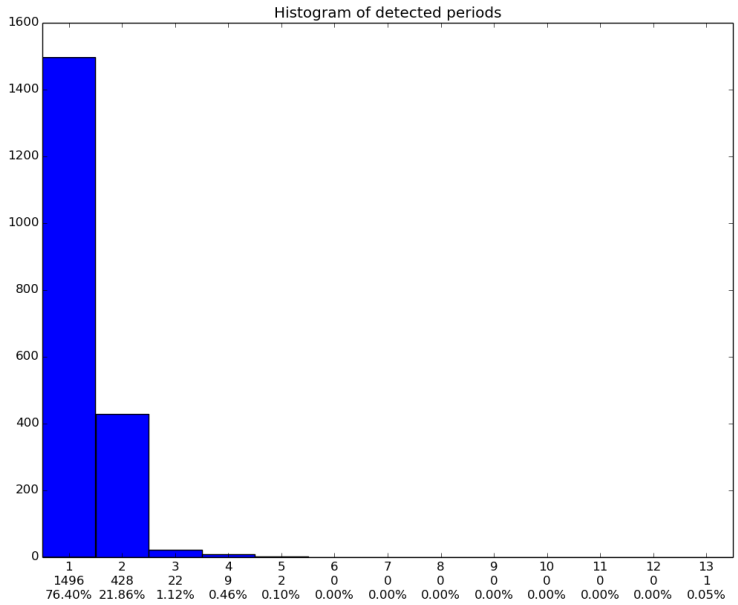

FIGURE 6.2. Histogram of the detected periods.

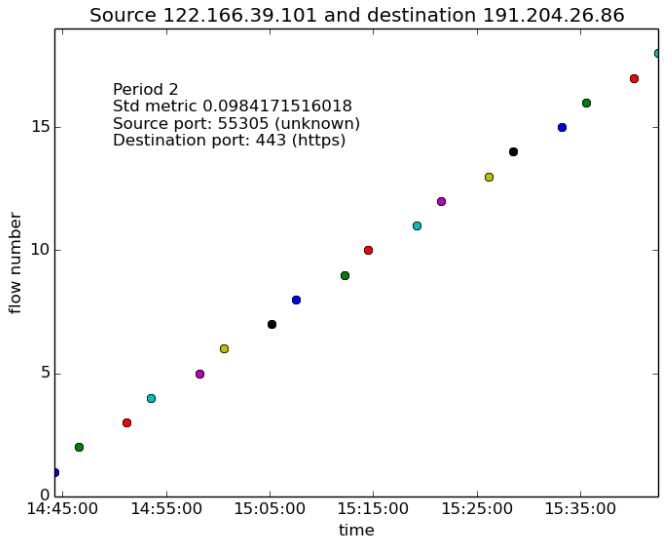

FIGURE 6.3. A pattern of period 2 in the data set.

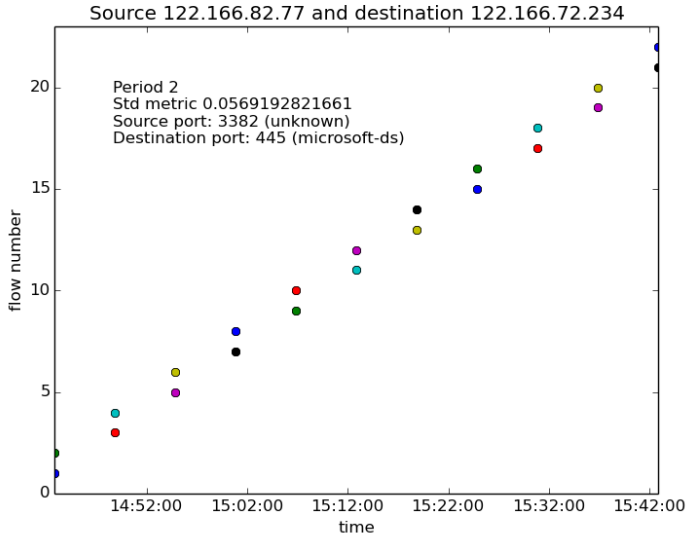

FIGURE 6.4. A pattern of period 2 in the data set.

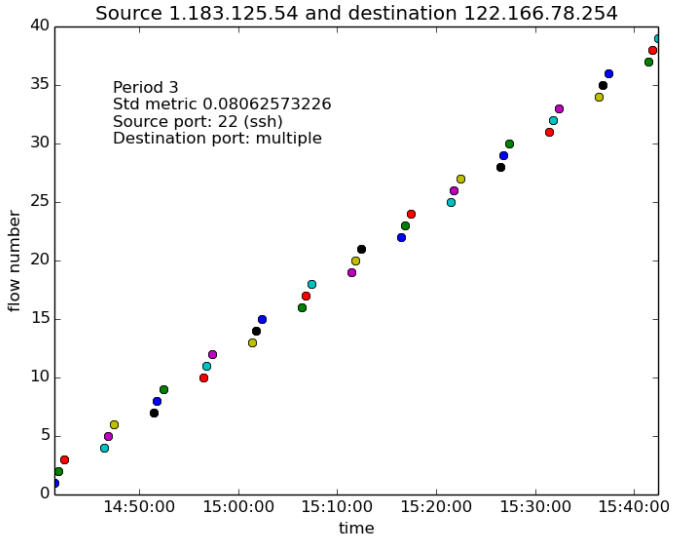

FIGURE 6.5. A pattern of period 3 in the data set.

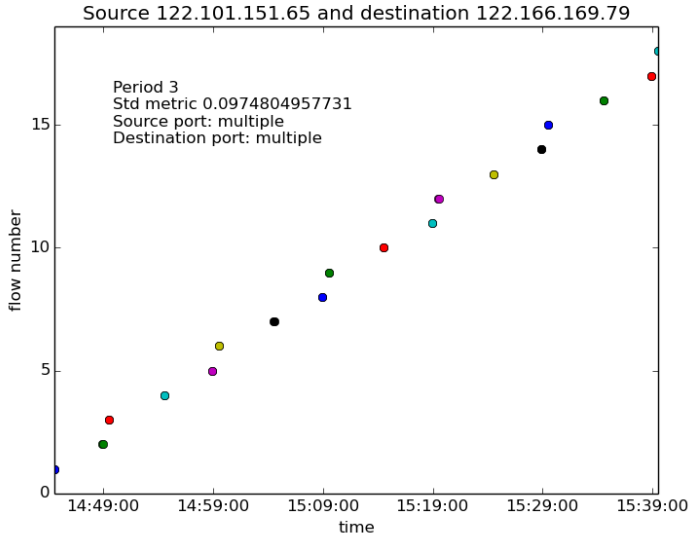

FIGURE 6.6. A pattern of period 3 in the data set.

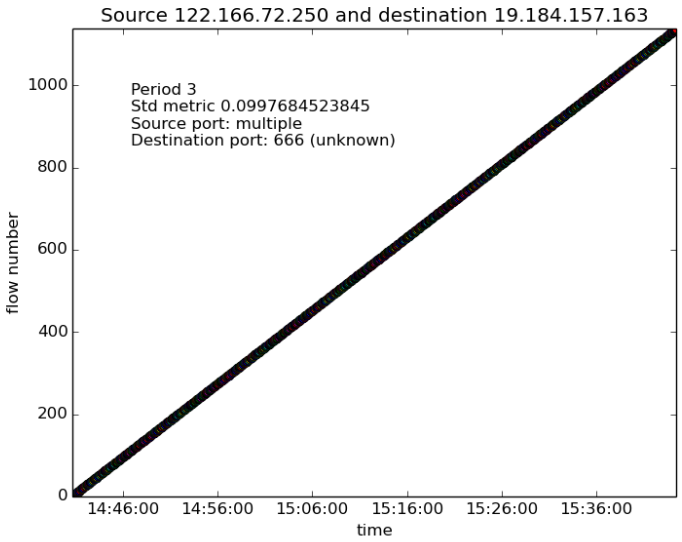

FIGURE 6.7. A pattern of period 3 in the data set. Not easy to see in the plot, the interarrival times alternate between 1, 3 and 6 seconds.

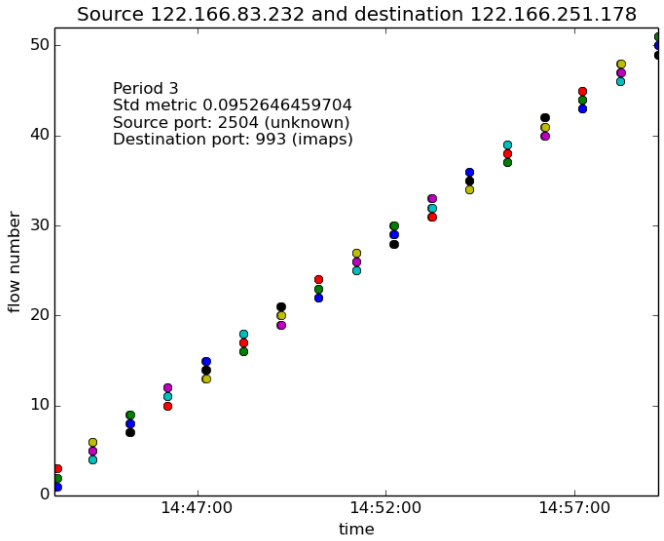

FIGURE 6.8. A pattern of period 3 in the data set.

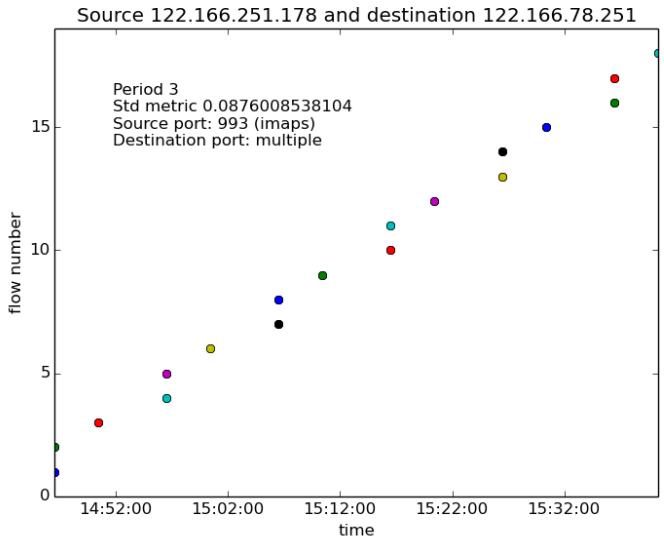

FIGURE 6.9. A pattern of period 3 in the data set.

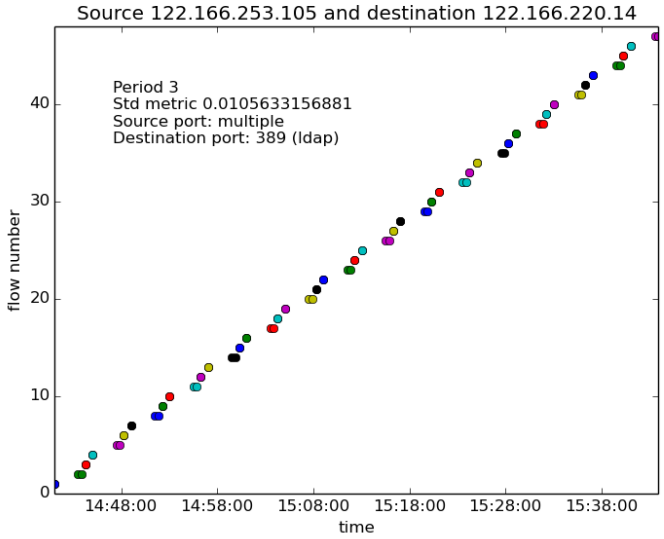

FIGURE 6.10. A pattern of period 3 in the data set.

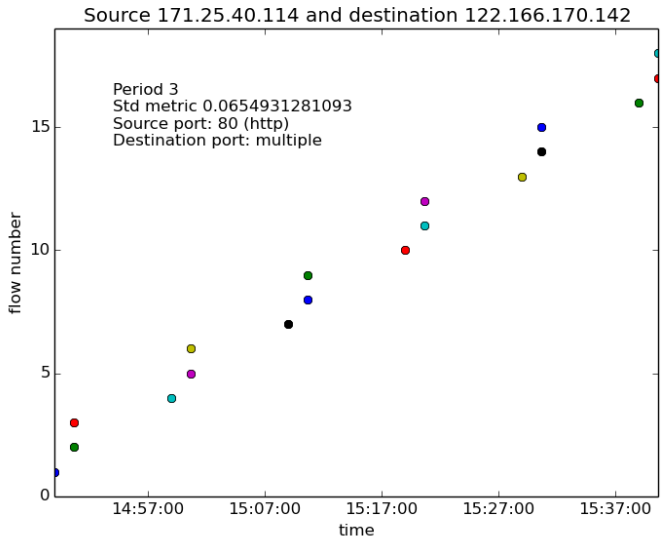

FIGURE 6.11. A pattern of period 3 in the data set.

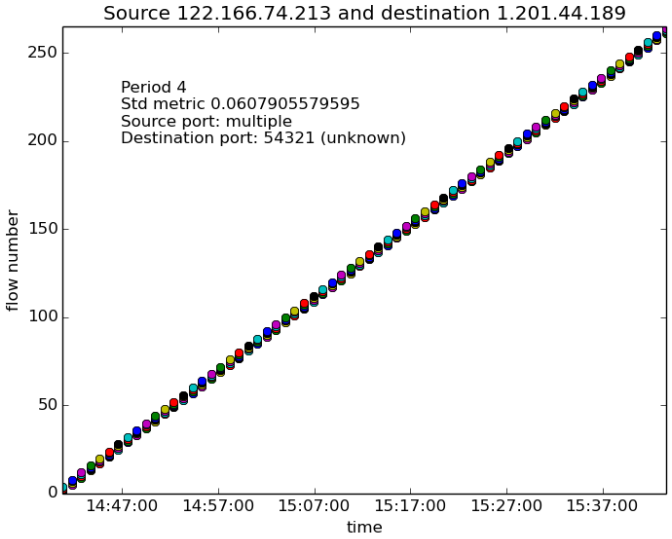

FIGURE 6.12. A pattern of period 4 in the data set.

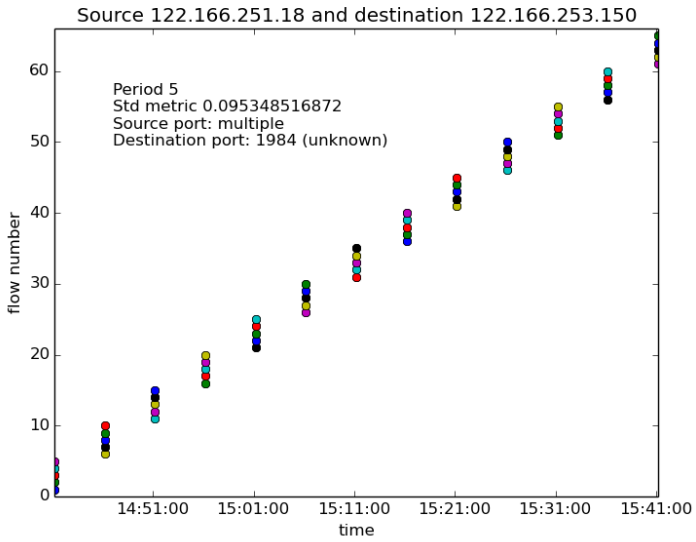

FIGURE 6.13. A pattern of period 5 in the data set.

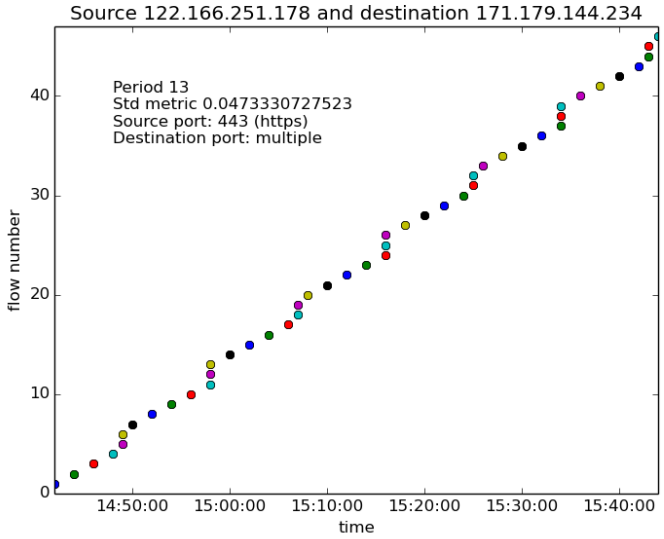

FIGURE 6.14. The pattern of period 13 in the data set.

6.7. Conclusions

Surprisingly, many connections exhibit periodicity with a period other than one. Not only patterns with period 2 were detected, but also a variety of patterns with higher periods. In existing work, it was not yet described that network traffic contains patterns which are more complex than the simple polling behaviour we already detected in Chapter 5. With the method developed in this chapter, we are able to efficiently find such connections in the data set, increasing the total number of periodic connections found by roughly a third, without finding false-positives.

CHAPTER 7

Experiment 3: Multi-dimensional data

7.1. Goal

So far, we only considered the timing information of flows to determine periodicity. However, our data set is much richer and also contains information on the duration and size of the flows and the number of packets of which the flow consists. To see if this can aid periodicity detection and to see whether regularities in interarrival times and other features coincide, we develop a general method of segmenting the data in parts. On these parts, we can then define metrics which take into account different features of the data.

7.2. Fixed segmentation

Segmentation is dividing the data into parts, segments. The segments can then be compared to one another, and if the segments are similar (their mutual differences are small) enough, the data is classified as periodic.

The trivial segmentation is to give every observation its own segment. A little more sophisticated is to segment the data into segments of the same size. When we later define a metric, the size p of the segments will be the candidate period for which we compute a score to determine if the data is periodic or not.

Indyk, Koudas and Muthukrishnan ([18], Section 2.3.4) divide a time series $V = (v_1, \ldots, v_n)$ with elements $v_i \in \mathbb{R}$ into segments of a fixed size. For our purposes, we wish to segment a sequence of k-dimensional data into fixed segments. Hence we introduce the following notation. For a sequence $V = (v_1, \ldots, v_n)$ with elements $v_i \in \mathbb{R}^k$, a fixed segmentation $V(p)$ of size p is a division of V into $m = \lfloor n/p \rfloor$ segments $V(p) = (s_1, \ldots, s_m)$, where each of the segments consists of p consecutive elements from V, $s_i = (v_{(i-1)p+1}, \ldots, v_{(i-1)p+p})$. (If n does not evenly divide into p, the last values of V are ignored.) The segments s_i are thus $p \times k$-dimensional vectors.

EXAMPLE 7.1. The following ordered sequence V consists of $n = 13$ elements which each have dimension $k = 3$.

$$\begin{aligned} V = [&(0, 9, 553), (60, 9, 553), (0, 5, 344), (60, 5, 344), (0, 9, 553), \\ &(60, 5, 344), (0, 9, 553), (60, 29, 2247), (0, 5, 344), (59, 9, 553), \\ &(0, 5, 344), (60, 9, 553), (0, 5, 344)]. \end{aligned}$$

A fixed segmentation with $p = 2$ divides V into $m = \lfloor 13/2 \rfloor = 6$ segments of length $p = 2$. Because $n = 13$ does not evenly divide into $p = 2$, the last element of V is discarded and the resulting segmentation is $V(2) = (s_1, \ldots, s_6)$, with

$$s_1 = [(0, 9, 553), (60, 9, 553)], \qquad s_2 = [(0, 5, 344), (60, 5, 344)],$$
$$s_3 = [(0, 9, 553), (60, 5, 344)], \qquad s_4 = [(0, 9, 553), (60, 29, 2247)],$$
$$s_5 = [(0, 5, 344), (59, 9, 553)], \qquad s_6 = [(0, 5, 344), (60, 9, 553)].$$

To define the similarity between the segments, we have to choose a metric to compute the (dis)similarity between segments. One way to do this is to find or compute a representative segment, which can then be compared to all the segments using some distance function, as discussed in the next section.

7.3. Average trend

Indyk, Koudas and Muthukrishnan [18] define the average trend as the segment that minimises the sum of distances to the other segments. In our notation, this is the segment s_j that minimises

$$D(V(p), s_j) = \sum_{i=1}^{m} d(s_i, s_j), \tag{7.1}$$

for all possible values of p and s_j, under a distance function d.

In [18], the vector V contains elements of $\mathbb{R}$, so the data is not multi-dimensional. However, a distance can intuitively be extended to multiple dimensions and the concept of finding an average trend that represents the pattern in the data is appealing. Therefore we choose to expand upon the idea of segmentation to research the periodic behaviour of our data set over multiple dimensions. To this end, we make several observations and adaptations w.r.t. Equation (7.1).

First off, note that Indyk, Koudas and Muthukrishnan look for an average trend *within* the set of all segments, which is cumbersome and computationally intensive. We propose instead to *compute* the average trend as a vector that has minimal summed distance to the segments, which may be different from all of the segments. So instead of computing Equation (7.1) for all possible segments, we compute one the minimising segment instead. For the case in Indyk et al., the distance function d used is the Euclidean distance. The average trend is thus the segment $s_j \in V(p)$ minimising

$$D(V(p), s_j) = \sum_{i=1}^{m} ||s_i - s_j|| \tag{7.2}$$

over all possible values of p and $s_j \in V(p)$. In this case, the minimiser for a given p is the geometric median. Though it is not possible to compute this in closed form in general, an approximation algorithm exists. (See Section 7.4.) If one chooses instead (as we propose) to use the squared Euclidean distance as distance function d, the sum of distances for a given p is minimised by the mean. (See Section 7.5.)

The second observation we make is that Equation (7.2) is susceptible to overfitting. Because the distance of the chosen average trend to itself is always zero, longer average trends tend to have a shorter distance. Using the squared Euclidean distance is advantageous here, as discussed in Section 7.6.

Finally, because we consider multi-dimensional data, adaptations need to be made to compare the different dimensions in a sensible way, since they represent different magnitudes and units. This is discussed in Section 7.7. From these observations and adaptations, we use the segmentation approach to derive a new metric for multi-periodic multi-dimensional periodicity detection (Sections 7.8 and 7.9), which turns out to be a generalisation of the metric in Experiments 1 and 2.

7.4. Geometric median

For points $s_1, \ldots, s_m$ in $\mathbb{R}^{p\times k}$, which we will call foci, a geometric median is a point $s \in \mathbb{R}^{p\times k}$ minimising the sum of Euclidean distances to the foci,

$$\sum_{i=1}^{n} ||s_i - s||. \tag{7.3}$$

The name 'geometric median' stems from the fact that for $p\times k = 1$, this minimum is attained by the median.

LEMMA 7.1. *Suppose $p \times k = 1$ and the metric is the Euclidean metric (as used by Indyk et al.) Then the median points and all points on a line between two median points are a geometric median.*

PROOF. Let the foci be ordered such that $s_1 \leq \cdots \leq s_m$ and suppose the point $s \in \{s_1, \ldots, s_m\}$ to which the foci are compared is at s_j, $j < m$. Moving s from s_j to s_{j+1} increases the distance of s to $s_1, \ldots, s_j$ by $s_{j+1} - s_j \geq 0$ and decreases the distance of s to $s_{j+1}, \ldots, s_m$ by the same amount. Thus the increase in total distance is $(2j - m) \cdot (s_{j+1} - s_j)$. For $j > \frac{m}{2}$ this increase is non-negative and for $j < \frac{m}{2}$, this increase is non-positive. If m is even, the increase is zero for $j = \frac{m}{2}$. Hence for even m,

$$f(s_1) \leq \cdots \leq f(s_{m/2}) = f(s_{m/2+1}) \geq \cdots \geq f(s_m).$$

For uneven m,

$$f(s_1) \leq \cdots \leq f(s_{\lceil m/2 \rceil}) \geq f(s_{\lceil m/2 \rceil + 1} \geq \cdots f(s_m).$$

Thus out of the s_i, the median points minimise total distance to the other points.

Let y be a point between s_1 and s_m, so that $s_j \leq y \leq s_{j+1}$ for some $j \in \{1, \ldots, m-1\}$. The j foci $s_1, \ldots, s_j$ are smaller than y and the $m - j$ foci

$s_{j+1}, \ldots, s_m$ are larger than y. Hence,

$$\begin{aligned} f(y) &= \sum_{i=1}^{m} ||s_i - y|| \\ &= (y - s_1) + \cdots + (y - s_j) \\ &+ (s_{j+1} - y) + \cdots + (s_m - y) \\ &= -\sum_{i=1}^{j} s_i + j \cdot y + \sum_{i=j+1}^{m} s_i - (m - j)y \\ &= \sum_{i=1}^{j} s_i - \sum_{i=j+1}^{m} s_i + (2j - m) \cdot y. \end{aligned}$$

Thus, $f(s_j) \leq f(y)$ if $2j - m \geq 0$ and $f(s_{j+1}) \leq f(y)$ if $2j - m \leq 0$. Suppose now that f is minimised by some $y \in \mathbb{R}$ which is different from all the foci s_i. If $y < s_1$, then the distance to all foci decreases by increasing y, so y cannot be minimal. Likewise, if $y > s_m$, then the distance to all foci decrease by decreasing y, so y cannot be minimal. Thus, $s_j < y < s_{j+1}$ for some $j \in \{1, \ldots, m-1\}$. Then $f(s_j) \leq f(y)$ holds if $2j - m \geq 0$ and/or $f(s_{j+1}) \leq f(y)$ holds if $2j - m \leq 0$. Thus the minimum is always attained by one of the median points (and, if $2j = m$, also by the points in-between). □

For $p \times k > 1$, there is a unique point minimising Equation (7.3), provided the foci do not all lie on one straight line. The proof of this statement, given below, is an alternative proof of the theorem in [42, 43].

THEOREM 7.1. *The geometric median is unique if the foci do not all lie on one line. Else, the median points on this line are geometric medians.*

PROOF. Let D' be a closed ball with centre M that contains all the foci $s_1, \ldots, s_m$ in its interior, let D be a strictly larger closed ball with the same centre that contains D' in its interior and let $\delta D'$ and δD be their boundaries. We will first show that a local minimum of $f : \mathbb{R}^{p \times k} \to \mathbb{R}$, $f(y) = \sum_{i=1}^{m} ||s_i - y||$ is attained in D because it is closed and bounded. This extreme-value theorem follows from some well-established properties of metric spaces (see e.g. [44]). Since D is a closed and bounded subset of $\mathbb{R}^{p \times k}$, it is compact (Heine-Borel theorem). Because D is compact and non-empty and f is a continuous function, the image $f(D)$ of f is also compact. As $f(D)$ is a compact subset of $\mathbb{R}$, it is closed and bounded (again Heine-Borel). Since $f(D)$ is not empty (it contains at least the images of the foci) and bounded, its infimum $m = \inf f(D)$ exists. Because $f(D)$ is compact it is complete, so that the infimum m lies in $f(D)$. Hence there is an $r \in D$ for which $f(r) = m$, the minimum valued attained by f on D.

The minimum of f in D could be attained on the boundary δD. To see that this is not the case, suppose for the sake of argument that the minimum of f is attained by $r \in \delta D$. Pick r' to be a point on the line from M to r such that r' lies

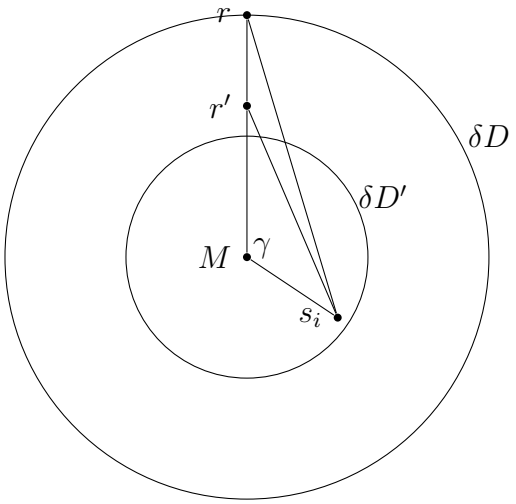

FIGURE 7.1. Sketch of the situation used to prove that the minimum of f is not attained on δD.

in the interior of D but not in D'. Take s_i to be an arbitrary focus. We will show that $||s_i - r'||$ is strictly smaller than $||s_i - r||$.

Note that $||M - r'|| > ||M - s_i||$ since s_i lies inside D while r' does not. We let γ be the angle between the lines from s_i to M and from r to M. The situation is sketched in Figure 7.1. We apply the law of cosines.

$$\begin{aligned}
||s_i - r||^2 &= ||s_i - M||^2 + ||M - r' + r' - r||^2 - 2||s_i - M||||M - r' + r' - r|| \cos\gamma \\
&= ||s_i - M||^2 + ||M - r'||^2 + ||r' - r||^2 + 2||M - r'||||r' - r|| \\
&\quad - 2||s_i - M||||M - r'|| \cos\gamma - 2||s_i - M||||r' - r|| \cos\gamma \\
&\geq \Big(||s_i - M||^2 + ||M - r'||^2 - 2||s_i - M||||M - r'|| \cos\gamma\Big) \\
&\quad + ||r' - r|| + 2||M - r'||||r' - r|| - 2||s_i - M||||r' - r|| \\
&= ||s_i - r'||^2 + ||r' - r||^2 + 2||r' - r||(||M - r'|| - ||s_i - M||) \\
&> ||s_i - r'||^2.
\end{aligned}$$

Hence the minimum of f in D is not attained on the boundary, so that there must be an r in the interior of D for which $f(r) = m$ is minimal in D. Then there exists a small open ball $B_\epsilon(r)$ of size $\epsilon > 0$ around r which lies completely in D. Hence, for all $y \in B_\epsilon(r)$, $f(r) \leq f(y)$, so that $f(r)$ is a local minimum.

Next we show that the local minimum $f(r)$ (which we know to be minimal in D) is a global minimum of f. Because of the triangle inequality, $f(y)$ is convex:

for $t \in [0,1]$ and $x, y \in \mathbb{R}^{p\times k}$,

$$\begin{aligned} f(tx+(1-t)y) &= \sum_{i=1}^{m} ||s_i - tx - (1-t)y|| \\ &= \sum_{i=1}^{m} ||t(s_i - x) + (1-t)(s_i - y)|| \\ &\leq \sum_{i=1}^{m} \left[||t(s_i - x)|| + ||(1-t)(s_i - y)|| \right] \\ &= \sum_{i=1}^{m} \left[t||s_i - x|| + (1-t)||s_i - y|| \right] \\ &= t\sum_{i=1}^{m} ||s_i - x|| + (1-t)\sum_{i=1}^{m} ||s_i - y|| \\ &= tf(x) + (1-t)f(y). \end{aligned}$$

Let $x \in \mathbb{R}^{p\times k}$ be arbitrary. If we take $t > 0$ small enough, $tx+(1-t)r$ will be close to r and thus lie inside D, so that $f(r) \leq f(tx+(1-t)r)$ by minimality of r. Then, because f is convex, $f(r) \leq f(tx + (1-t)r) \leq tf(x) + (1-t)f(r)$. Rearranging terms, $f(r) - (1-t)f(r) \leq tf(x)$ or $f(r) \leq f(x)$. Since x was arbitrary, $f(r)$ is a global minimum.

It remains to be shown that the minimum is unique. Suppose both y and y' minimise f and $y \neq y'$. We will show that in this case, the s_i all lie on the line $L = \{y' + \lambda(y - y') | \lambda \in \mathbb{R}\}$. By convexity, $f(\frac{1}{2}y + \frac{1}{2}y') \leq \frac{1}{2}f(y) + \frac{1}{2}f(y') = m$, so that by the minimality of m, $f(\frac{1}{2}y + \frac{1}{2}y') = m$. By the triangle inequality,

$$\begin{aligned} m &= \sum_{i=1}^{m} ||s_i - \frac{1}{2}y - \frac{1}{2}y'|| \\ &= \sum_{i=1}^{m} ||\frac{1}{2}(s_i - y) + \frac{1}{2}(s_i - y')|| \\ &\leq \sum_{i=1}^{m} \left[||\frac{1}{2}(s_i - y)|| + ||\frac{1}{2}(s_i - y')|| \right] \\ &= \frac{1}{2}\sum_{i=1}^{m} ||s_i - y|| + \frac{1}{2}\sum_{i=1}^{m} ||s_i - y'|| \\ &= m, \end{aligned}$$

so that the triangle inequality $||\frac{1}{2}(s_i - y) + \frac{1}{2}(s_i - y')|| \leq ||\frac{1}{2}(s_i - y)|| + ||\frac{1}{2}(s_i - y')||$ must be met with equality for all $i = 1, \ldots, m$.

For points a and b, if $||a + b|| = ||a|| + ||b||$, then the triangle consisting of a, b and $a + b$ is degenerate and a, b and $a + b$ lie on one line. Thus all points are on the line $\{a + \lambda(b - a) | \lambda \in \mathbb{R}\}$. For $a + b$ we then find $a + b = a + \lambda(b - a)$ for some λ, or equivalently $\lambda a = (\lambda - 1)b$. This equation is satisfied by $a = 0$ when $\lambda = 1$,

$b = 0$ when $\lambda = 0$ or $a = (\lambda - 1)/\lambda b$ when $\lambda \neq 0, 1$. Thus if $||a|| + ||b|| = ||a + b||$, then either $a = 0$, $b = 0$ or $a = \alpha b$ for some scaling factor $\alpha \neq 0$.

We have found that $||\frac{1}{2}(s_i - y) + \frac{1}{2}(s_i - y')|| = ||\frac{1}{2}(s_i - y)|| + ||\frac{1}{2}(s_i - y')||$ for all $i = 1, \ldots, m$. Thus for any arbitrary i, $1 \leq i \leq m$, either $\frac{1}{2}(s_i - y) = 0$, $\frac{1}{2}(s_i - y') = 0$ or $\alpha_i \frac{1}{2}(s_i - y) = \frac{1}{2}(s_i - y')$ for some scaling factor $\alpha_i \neq 0$. In the first case, $s_i = y$ and s_i lies on L (take $\lambda = 1$). In the second case, $s_i = y'$ and s_i lies on L (take $\lambda = 0$). In the last case, rewriting gives $s_i = y' + \frac{\alpha_i}{1-\alpha_i}(y' - y)$, which also lies on L (take $\lambda = \frac{\alpha_i}{1-\alpha_i}$). Thus if there are two minima, all s_i lie on the line L. By Lemma 7.1 the minimum is then attained by one of the median points on the line. Else, if the foci do not all lie on one line, the minimum is unique. □

There is no closed-form expression for the geometric median, but there is an algorithm to derive an approximation converging to the geometric median [45], which alludes to the first of three proofs in [42]. The algorithm is based on taking an arbitrary starting point y_0 and defining

$$y_{n+1} = \sum_{i=1}^{m} \frac{s_i}{||s_i - y||} / \sum_{i=1}^{m} \frac{1}{||s_i - y_n||}, \qquad n \in \mathbb{N}. \tag{7.4}$$

As long as y_n is never equal to one of the foci, y_{n+1} exists and y_n converges to the minimum. In [45], the algorithm is slightly modified to avoid y_n becoming one of the foci, so that $\{y_n\}$ always converges to a solution.

7.5. Squared distances

Another natural candidate for the distance function d in Equation (7.1) is the squared Euclidean distance. As we will show, this distance function has the benefit that it is minimised by the mean. The mean is trivial to compute for a given segmentation size and in this way avoids having to find the minimising segment.

For a set of points or segments $s_1, \ldots, s_m$ in $\mathbb{R}^{p \times k}$, the sum of squared distances to a segment or point $y \in \mathbb{R}^d$ is

$$f(y) = \sum_{i=1}^{m} ||s_i - y||^2. \tag{7.5}$$

LEMMA 7.2. *The sum of squared distances in Equation 7.5 is uniquely minimised by the mean.*

PROOF. Let $s_1, \ldots, s_m \in \mathbb{R}^{p \times k}$, with mean $s = \frac{1}{m} \sum_{i=1}^{m} s_i$ and let $y \in \mathbb{R}^{p \times k}$ be arbitrary. Denote the $p \times k$ elements of s and the s_i as $s(j)$ and $s_i(j)$, $1 \leq j \leq p \times k$.

Then,

$$\begin{aligned}
f(y) &= \sum_{i=1}^{m} ||s_i - y||^2 \\
&= \sum_{i=1}^{m} ||(s_i - s) + (s - y)||^2 \\
&= \sum_{i=1}^{m} \sum_{j=1}^{p\times k} [(s_i(j) - s(j)) + ((s(j) - y(j))]^2 \\
&= \sum_{i=1}^{m} \sum_{j=1}^{p\times k} (s_i(j) - s(j))^2 \\
&+ 2\sum_{i=1}^{m} \sum_{j=1}^{p\times k} (s(j) - y(j))(s_i(j) - s(j)) \\
&+ \sum_{i=1}^{m} \sum_{j=1}^{p\times k} (s(j) - y(j))^2.
\end{aligned}$$

The terms in the first summation do not include y and are thus constant. Rearranging the terms in the second summation, we find that

$$\begin{aligned}
\sum_{i=1}^{m} \sum_{j=1}^{p\times k} (s(j) - y(j))(s_i(j) - s(j)) &= \sum_{j=1}^{p\times k} (s(j) - y(j)) \sum_{i=1}^{m} (s_i(j) - s(j)) \\
&= \sum_{j=1}^{p\times k} (s(j) - y(j)) \cdot 0 \\
&= 0.
\end{aligned}$$

Thus the only part of $f(y)$ which depends on y is the last summation,

$$\sum_{i=1}^{m} \sum_{j=1}^{p\times k} (s(j) - y(j))^2 = m \cdot \sum_{j=1}^{p\times k} (s(j) - y(j))^2.$$

This part is non-negative because it is a sum of squares and it is equal to zero if and only if $y = s$. Hence, $f(y)$ is uniquely minimised by the mean. □

7.6. Overfitting

In the segmentation score $D(V(p), s_j) = \sum_{i=1}^{m} d(s_i - s_j)$ of Equation (7.1), the j-th term of the sum, $||s_j - s_j||$, always equals zero. Thus, one of the m segments always equals zero. If the segmentation length p is greater, the number of segments m is lower and a relatively larger part of the terms in the segmentation score equals zero. This may lead to overfitting, as the following example shows.

EXAMPLE 7.2. Let $V = (61, 28, 59, 31, 60, 29, 61, 31, 58, 30)$, a repetition of the pattern $(60, 30)$ with some noise. We use the Euclidean metric. For a segmentation size of $p = 2$, there are 5 segments and $D(V(2), s_j)$ is minimised by $s_3 = (60, 29)$ with a score of 8.1. For $p = 4$, there are only two segments. One of those contributes a score of 0 so that the score is based on one segment of length four. The last two values of V are cut off because they do not fit in a full segment. The minimiser of $D(V(4), s_j)$ is $s_1 = (61, 28, 59, 31)$ with a segmentation score of 2.4.

A helpful first step in solving this problem is to minimise not the total, but the average distance to other segments. Since s_j always has distance zero to itself, we divide by $m - 1$ instead of m. Instead of minimising Equation (7.2), we then find a segment s_j with period p minimising the average distance to the other segments,

$$D'(V(p), s_j) = \frac{1}{m-1} \sum_{i=1}^{m} d(s_i, s_j), \tag{7.6}$$

with the restriction that p must be chosen such that there are at least $m \geq 2$ segments.

When d is the Euclidean distance, this segmentation score still favours larger segments, because the square root in the Euclidean distance is a concave function.

When d is the squared Euclidean distance, the distance $d(s_i, s_j)$ consist of the sum of the squared element-wise differences. Of course, $d(s_i, s_i) = 0$, but for the other i, $d(s_i, s_j)$ is the sum of $p \times k$ squared element-wise differences. Since $d(s_i, s_j)$ is also summed, this gives a total of $(m-1)p \times k$ squares, with $p = \lceil n/k \rceil$, so that this equals about $(m-1)n$. Because of the division by $m - 1$, the factor $m - 1$ cancels out. Thus, there is no longer a dependence on p in this case and no overfitting of larger periods occurs, making it beneficial to use the squared Euclidean distance with the modified segmentation score.

EXAMPLE 7.3. Let $V = (61, 28, 59, 31, 60, 29, 61, 31, 58, 30)$ as in Example 7.2. With Equation (7.6), we have for the Euclidean distance that $D'(V(2), s_j)$ is minimised by $s_3 = (60, 29)$ with score 2.03 and $D'(V(4), s_j)$ is minimised by $s_1 = (61, 28, 59, 31)$ with score 2.4. For the squared Euclidean distance, the same minimisers are found with scores of 4.3 and 6 respectively.

7.7. Scaling

The data in the dimensions of the data may consist of different quantities, with different units, thus making the magnitudes incomparable; e.g. in our case an observation may have two dimensions: an interarrival time (quantity) of 60 (magnitude) seconds (unit) and a flow size (quantity) of 553 (magnitude) bytes (unit). The units impose a certain level of arbitrariness on the magnitude; e.g. if the interarrival time were measured in milliseconds instead of seconds its magnitude would increase a thousand-fold and if the flow size were measured in bits

instead of bytes its magnitude would increase by a factor eight. Because interarrival time and flow size cannot be expressed in the same unit, this makes direct comparison between their magnitudes an arbitrary and therefore unjustified choice. Thus, it is sensible to scale each dimension by a scaling factor.

This means that we will have to divide the values $v_i(j)$ by a scaling factor. We could choose a fixed scaling factor α_j for every dimension $j = 0, \ldots, k-1$, but that would require experimentally finding (or guessing) k values. Another approach is to normalise each element of the segment, choosing a scaling factor $\alpha_j = \frac{1}{m}\sum_{i=1}^{m} s_i(j)$. However, this has the disadvantage that it is difficult to compare different periods. Instead we choose α_j to be the mean value of V in that dimension, $\alpha_j = \frac{1}{n}\sum_{i=1}^{n} v_i(j)$.

7.8. Combination and reconciliation with Experiment 2

We will be using the squared Euclidean distance and the adapted segmentation score from Equation (7.6), because it does not overfit. From Section 7.5 we know that instead of searching for an s_j minimising Equation (7.6) we can also, for a given p, compute the mean and use this as a minimiser. For a given p, the distance function then becomes

$$D'(V(p)) = \frac{1}{m-1}\sum_{i=1}^{m} d(s_i, \overline{s}). \tag{7.7}$$

We choose to keep the denominator at $m-1$ instead of m even though in contrast with Equation (7.6) a comparison is now made with all m segments. In analogy with the sample variance, one degree of freedom is used in calculating $\overline{s}$ and in this way the segmentation score is in a way an extension of the sample variance, with a clear relation to the metric used in Experiment 2, as will be shown later.

Recall that the segments s_i are $p \times k$-dimensional vectors, where p is the segment length and k the dimension of the data. The mean $\overline{s}$ averages the segments s_i, $\overline{s} = \frac{1}{m}\sum_{i=1}^{m} s_i$ and thus we have

$$\overline{s} = \begin{pmatrix} \overline{v}_{0,0} & \overline{v}_{0,1} & \cdots & \overline{v}_{0,k-1} \\ \vdots & \vdots & & \vdots \\ \overline{v}_{p-1,0} & \overline{v}_{p-1,1} & \cdots & \overline{v}_{p-1,k-1} \end{pmatrix},$$

where $\overline{v}_{j,l}$ denotes the average over the segments, $\overline{v}_{j,l} = \frac{1}{m}\sum_{i=0}^{m-1} v_{ip+j,l}$.

The squared Euclidean distance is given by the squares of the differences of every component. We thus have

$$d(s_i, \overline{s}) = \sum_{l=0}^{k-1}\sum_{j=0}^{p-1}(v_{ip+j,l} - \overline{v}_{j,l})^2,$$

which gives an adapted segmentation score

$$\begin{aligned} D'(V(p)) &= \frac{1}{m-1}\sum_{i=0}^{m-1}\sum_{l=0}^{k-1}\sum_{j=0}^{p-1}(v_{ip+j,l}-\overline{v}_{j,l})^2 \\ &= \sum_{l=0}^{k-1}\sum_{j=0}^{p-1}\frac{1}{m-1}\sum_{i=0}^{m-1}(v_{ip+j,l}-\overline{v}_{j,l})^2, \end{aligned}$$

which is a (double) sum of sample variances. The sample variance is taken of the elements of V which have position j in each segment, at dimension l.

If we take the dimension to be $k = 1$, we see that this the score is

$$D'(V(p)) = \sum_{j=0}^{p-1}\frac{1}{m-1}\sum_{i=0}^{m-1}(v_{ip+j}-\overline{v}_j)^2,$$

which is similar to Equation (6.7) of Experiment 2,

$$s = \sqrt{\sum_{i=1}^{k}\frac{1}{m_i-1}\sum_{j=0}^{m_i-1}(d_{i+jp}-\overline{d_i})^2}. \tag{6.7}$$

Apart from the square root, the difference is that we cut off the values of V after segment $m = \lceil n/p \rceil$, in keeping with Indyk's definition. Now that we have framed $D'(V(P))$ as a sum of variances, we might just as well add the values that were cut off back in again, to yield

$$\begin{aligned} D''(V,p) &= \sum_{l=0}^{k-1}\sum_{j=0}^{p-1}\text{SampleVar}(\{v_{i,l}|i=j \pmod p\}) \\ &= \sum_{l=0}^{k-1}\sum_{j=0}^{p-1}\frac{1}{m_j-1}\sum_{i=0}^{m_j-1}(v_{ip+j,l}-\overline{v}_{j,l})^2. \end{aligned} \tag{7.8}$$

Since adding an extra dimension shouldn't increase the score, we divide by k. Furthermore, we take the square root of the score to further increase the similarity to the metric in Experiment 2. The following is our final periodicity score *without scaling*.

$$D'''(V,p) = \sqrt{\frac{1}{k}\sum_{l=0}^{k-1}\sum_{j=0}^{p-1}\frac{1}{m_j-1}\sum_{i=0}^{m_j-1}(v_{ip+j,l}-\overline{v}_{j,l})^2}. \tag{7.9}$$

THEOREM 7.2. *Our final periodicity score without scaling is a generalisation of the scores in Experiments 1 and 2.*

PROOF. If we reduce the number of dimensions to $k = 1$, Equation (7.9) reduces to

$$D'''(V,p) = \sqrt{\sum_{j=0}^{p-1} \frac{1}{m_j - 1} \sum_{i=0}^{m_j-1} (v_{ip+j} - \overline{v}_j)^2}. \tag{7.10}$$

Except for the different letters for the variables used, this is equal to Equation (6.7), which is the metric used in Experiment 2. If we further let $p = 1$, this reduces to

$$D'''(V,p) = \sqrt{\frac{1}{m-1} \sum_{i=0}^{m-1} (v_i - \overline{v})^2}, \tag{7.11}$$

which is the metric from Equation (5.1) used in Experiment 1. □

7.9. Periodicity score

Because we use multiple dimensions, we will have to scale Equation (7.9). As explained in Section 7.7, we choose the mean value of V in a given dimension, $\alpha_j = \frac{1}{n}\sum_{i=1}^{n} v_i(j)$, as the scaling factor for the values of V in that dimension. The periodicity score of a connection V for a period p is then given by

$$S(V,p) = \sqrt{\frac{1}{k}\sum_{l=0}^{k-1}\sum_{j=0}^{p-1} \frac{1}{m_j - 1} \sum_{i=0}^{m_j-1} (\frac{v_{ip+j,l} - \overline{v}_{j,l}}{\overline{v}_{,l}})^2}, \tag{7.12}$$

where $\overline{v}_{j,l}$ is the average of $v_{ip+j,l}$ over $i = 0, \ldots, m_j - 1$ and $\overline{v}_{,l}$ is the average of $v_{i,l}$ over $i = 0, \ldots, n-1$. It can be seen as a multi-dimensional, multi-period version of the coefficient of variation. A low score is an indicator of periodicity.

7.10. Method

Like in Experiments 1 and 2, we filter connections of fewer than 10 flows or lasting less than 60 seconds. For every connection with interarrival times, duration, number of packets and number of bytes stored in a vector V, the periodicity score $S(V,p)$ is then calculated for $p = 1$ up until $p = 5$ and the minimum of these scores is taken to be a connection's score. The amount of periods is limited, because of the computational complexity. We choose to calculate the periodicity score for $p = 1, \ldots, 5$ because in Experiment 2, all but one connection had a period of at most five. An alternative would be to calculate periods up to $p = \lceil n/10 \rceil$, where n is the number of flows in a connection. This allows larger periods to be detected, but prevents the sample variance from being calculated over less than ten samples. The computing time this approach would take is significantly larger.

Thus, a connection V, containing information on the interarrival times, duration, bytes and number of packets of the flows, will be called periodic if it contains at least 10 flows, has a minimum length of 60 seconds and its score in

Equation (7.12) is below a threshold τ, for which we will seek a suitable value in Section 7.12.

7.11. Implementation

As before, the implementation is done in Python using the Pandas and NumPy libraries. For $p = 1, \dots 5$, variances have to be computed for four dimensions which each have p different sample variances of length $\sim n/p$. The theoretical complexity is thus only $\sum_{p=1}^{5} \mathcal{O}(4p \cdot n/p) = \mathcal{O}(n)$. (If we were to choose p periods, the complexity would be $\mathcal{O}(p \cdot n)$.) However, because of the size of the data set, the twenty-fold increase from the five values of p and four dimensions and the extra computations needed to compute more variances, the actual computing time greatly increases compared to that of both Experiments 1 and 2. Note that Experiment 2 would also be of complexity $\mathcal{O}(n)$ if we only allowed $p = 1, \dots, 5$ as possible periods. A possible extension of the method lies in trying to find a method which guesses a candidate period, like we did in Experiment 2. For now, we'll stick with periods 1 to 5 and see what results we can get in this fashion.

7.12. Results

After calculating the scores, we still have to decide on a suitable threshold. In order to do this, we analyse the scores of connections that were labelled periodic in Experiment 2 to the scores of all connections. We have already found connections that are periodic in Experiment 2, at least when looking only at their interarrival times. For this analysis, scatter plots are given in Figures 7.2 and 7.3 and a table with the number of periodic connections for various thresholds is given in Table 7.1.

TABLE 7.1. The number of connections with a score below various powers of 10, for all connections and for those connections that were periodic in Experiment 2.

Threshold	All		Periodic in exp. 2	
	Absolute	Relative (%)	Absolute	Relative(%)
10	104886	99.97	1955	99.9
1	29450	28.1	1612	82.3
0.1	2334	2.2	941	48.1
0.01	1191	1.1	795	40.6
0.001	903	0.9	752	38.4
0.0001	271	0.26	256	13.1
0.00001	42	0.04	40	2.0
0.000001	6	0.0046	6	0.3

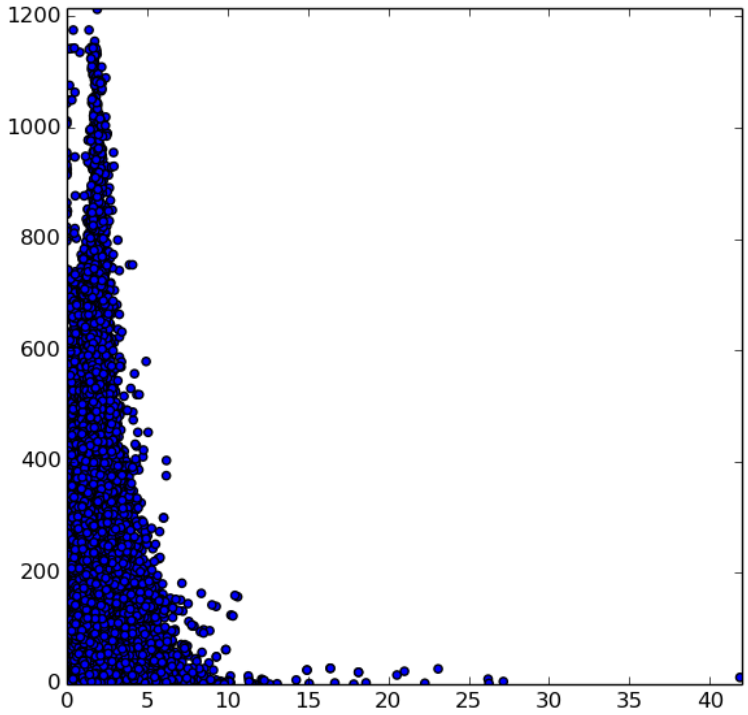

FIGURE 7.2. Scatter plot of the scores in Experiment 2 (horizontal axis, in seconds) and Experiment 3 (vertical axis, no unit).

Not surprisingly, the periodic connections of Experiment 2 generally have low scores in this Experiment. From Table 7.1 we see that there are $271 - 256 = 15$ connections with a very low score below 0.0001 which are not detected in Experiment 2. As an example, three of these connections are given in Appendix A. All three are detected in Experiment 3 as being periodic with period 5, which is too short relative to the number of flows ($n = 12$) to be detected by the local peaks method of Experiment 2. The other 12 connections not shown in the appendix also have very few ($n \sim 10$) flows. From Table 7.1 and Figure 7.2, we also see that some periodic connections from Experiment 2 have a high score in Experiment 3. Inspection of these outliers with score above 6 shows that they have outliers in the duration, bytes or number of packets, resulting in a higher score than for Experiment 2, where only interarrival times are taken into account. Because these outliers consist of many flows ($n \sim 1200$), we do not include them here or in an appendix.

Based on Table 7.1, we choose a threshold of 0.01. With this threshold, 1195 connections are periodic, with the periods distributed as shown in Table 7.2. If we look at the different dimensions separately, shown in Table 7.3, there seems to

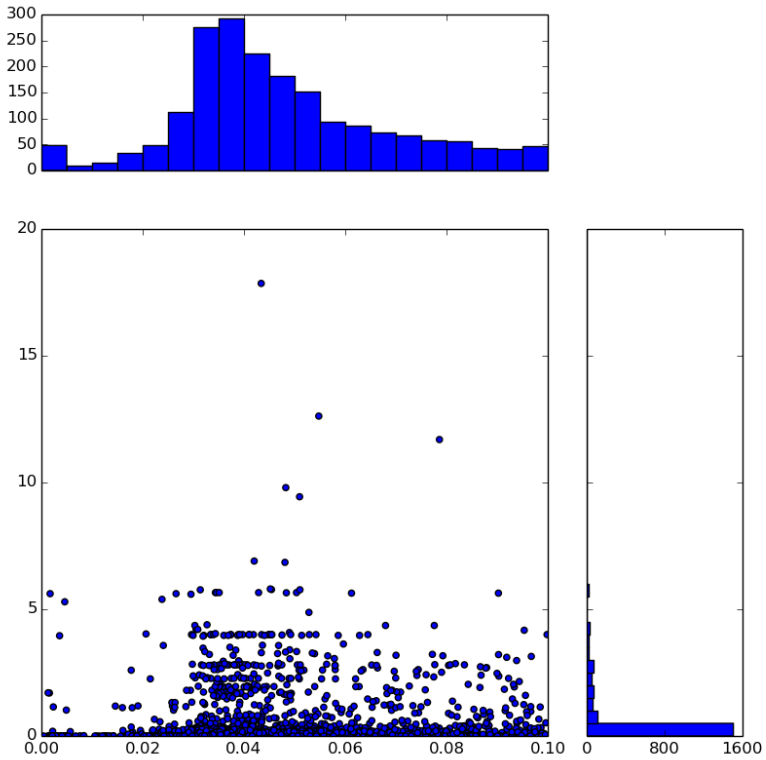

FIGURE 7.3. Scatter plot and histograms of the scores in Experiment 2 (horizontal axis, in seconds) and Experiment 3 (vertical axis, no unit) for connections that were detected as periodic in Experiment 2, thus having a score on the horizontal axis below 0.1 seconds.

be more periodicity in the dimensions of bytes and packets less periodicity in the dimension of duration than in the dimensions of interarrival time.

TABLE 7.2. Detected period of periodic connections (score below threshold of 0.01).

Period	# Connections
1	784
2	163
3	40
4	105
5	103
Total	1195

TABLE 7.3. Connections with score below threshold of 0.01 per dimension.

Dimension	# Connections
Interarrival time	6393
Duration	1503
Bytes	11546
Packets	11903

We manually looked into the connections that have a low score in Experiment 3, but weren't detected as periodic in Experiment 2 and the connections that were labelled to be periodic in Experiment 2 but were not detected in Experiment 3. There are 1017 connections that are periodic in Experiment 2 but not in Experiment 3 and 1393 connections periodic in Experiment 3 but not in Experiment 2. These connection do not lend themselves well for a graphic representation because the data is four-dimensional, but from manual inspection of the raw data of some of these connections, we learned that they fail to be periodic in one of the Experiments because they contain an outlier that only influences the score in one of the Experiments sufficiently to cross the threshold.

7.13. Conclusion

We made several adaptations to the segmentation technique by Indyk et. al., expanding it to be able to study at multi-dimensional data in a way that generalises our methods in Experiments 1 and 2, while resolving the problem of overfitting and reducing the complexity from $\mathcal{O}(n^2 \cdot p)$ to $\mathcal{O}(n \cdot p)$ when considering p periods.

From the data set, we learn that this method performs well in detecting periodicity. It does not seem to perform significantly worse or better than the method used in Experiment 2. This means that we now have two methods which work well to detect periodic connections. If this is called for, they can be used together to find a large number of periodic connections.

There is room for improvement of the method to choose appropriate candidate periods. Ideally, this should be done based on the length of the connection or on a period finder like in Experiment 2.

CHAPTER 8

Review and recommendations

8.1. Goal

The goal of this thesis was to develop periodicity detection techniques for netflow data and apply them to our data set of anonymized netflows from a computer network with approximately 3,000 hosts (described in Chapter 3). In order to do this, we first had to explore the concept of periodicity (Chapter 1) and review and categorise the existing techniques and literature (Chapter 2 for a general taxonomy of periodicity and Chapter 4 specifically for periodicity in network traffic). In the Experiments in Chapters 5 to 7, several detection techniques are applied to our data set. This chapter reviews the success of the three Experiments, outlines the differences between them and gives recommendations on the usage of the methods and on possible further research.

8.2. Recap

8.2.1. Experiment 1. In Experiment 1, we implemented the sample standard deviation technique on interarrival times by Hubballi & Goyal. Because we applied the technique to a data set containing traffic of thousands of computers instead of one, we found that a large number of false positives hinders the detection. We adapted the method so that it correctly detects periodic traffic without false positives and we showed the existence of many such connections in our data set.

The (adapted) method is easy to implement, accurately detects periodic connections and runs in $\mathcal{O}(n)$ time. It should be noted that the detection of a connection is dependent on the chosen time window and that connections that are periodic with a period larger than one are not detected.

8.2.2. Experiment 2. Since the method of Experiment 1 failed to detect connections with a period larger than one, we generalised the metric used in Experiment 1 to general periods. The new metric can be seen as multi-period extension of the sample standard deviation metric in Experiment 1. Combined with a method of finding candidate periods effectively, this allows us to increase the number of periodic connections found by 25% without introducing false positives. It also shows that a significant number of the periodic connections in real-life network traffic have patterns that are more sophisticated in nature than a simple

polling behaviour. This observation has, to the best of my knowledge, not been made in the existing literature.

The second method is still accurate, finds more connections than the method in Experiment 1 and is still computationally efficient with complexity $\mathcal{O}(n \log n)$.

8.2.3. Experiment 3. The method in Experiment 3 incorporates other aspects of the netflows besides the interarrival times, specifically the duration, number of bytes and number of packets. The basic approach is to divide the flows into different segments, which are compared to one another. Several improvements are made to this basic approach. By computing the best-fitting segment instead of looking for one, complexity is reduced by $\mathcal{O}(n)$. Furthermore, we motivate the use of the squared Euclidean distance, which allows easy computation and reduces overfitting issues. This leads to a new, multi-dimensional and multi-periodic method, which can be seen as a generalisation of Experiment 1 and 2. Because multiple dimensions with different units are compared, we introduce a relative scaling factor. The resulting measure resembles a sample coefficient of variation which has been made multi-periodic and multi-dimensional.

The method finds different data than that of Experiment 2, leaving out some connections and finding other highly regular connections instead. It is thus a good companion to the previous method.

8.3. Recommendations

To find periodic connections in netflow data, I recommend to use the method of Experiment 2, since it is easier to implement and faster to run than the method of Experiment 3, but finds more connections than the method of Experiment 1. If implementation of multiple methods is not an issue, it is wise to also implement the method of Experiment 3 and combine the results.

In all methods, the chosen time window is of importance, since small windows may leave out periodic connections and large windows have a larger chance to contain an outlier which skews the results. One interesting choice in this regard would be to not set a fixed window based on time, but to calculate scores when a certain number of flows has been reached. This could also be interesting when one wishes to make an online implementation, as the score could be (re)calculated every time this number of flows has been observed. Another approach to reduce the effect of outliers could be to filter a fixed percentage (say 5%) of the greatest outliers from the data before calculating scores.

The detection of malware was a motivation for this work, but is itself out of scope. Because the data set is anonymous, we only know that periodic traffic indicates machine-generated traffic. Applying our techniques to a non-anonymous data set would give insight in the processes behind the newly-found multi-period patterns found in Experiments 2 and 3. As a first step in detecting command and control traffic, I suggest to make a graph of the periodic connections to see

the 'hotspots' that have many periodic connections. Together with a reverse DNS lookup to find the domain names associated with computers, this could give a quick first impression on the sources of the periodic connections and whether they're likely to be malign or benign.

8.4. Conclusion

We were successful in finding a method to detect periodic connections in a large netflow data set. Furthermore, we found periodic connections with periods larger than one, which were not described in literature before, and developed a novel technique that takes into account the several dimensions of netflow data.

Depending on what one wants to detect and the implementation and computing time available, three different methods are presented, which each have distinctive advantages but are also logical extensions of one another. Together, they allow us to successfully determine which connections in our data set are periodic.

APPENDIX A

Several connections detected in Experiment 3

This appendix contains three connections detected in Experiment 3 and discussed in Section 7.10. Shown are the source and destination IP address (sip and dip) and port numbers (sp and dp), the protocol (in all cases 6, for TCP traffic), the TCP flags, number of packets (pkt), bytes/octets (oct), start and end time of the flow and the duration of the flow in seconds.

A.1. First example

sip	dip	sp	dp	proto	flags	pkt	oct
122.166.71.185	122.166.35.154	139	1321	6	0	1	46
122.166.71.185	122.166.35.154	139	1321	6	0	1	46
122.166.71.185	122.166.35.154	139	1321	6	0	1	46
122.166.71.185	122.166.35.154	139	1321	6	0	1	46
122.166.71.185	122.166.35.154	139	1321	6	0	1	46
122.166.71.185	122.166.35.154	139	1321	6	0	1	46
122.166.71.185	122.166.35.154	139	1321	6	0	1	46
122.166.71.185	122.166.35.154	139	1321	6	0	1	46
122.166.71.185	122.166.35.154	139	1321	6	0	1	46
122.166.71.185	122.166.35.154	139	1321	6	0	1	46
122.166.71.185	122.166.35.154	139	1321	6	0	1	46
122.166.71.185	122.166.35.154	139	1321	6	0	1	46

start	end	duration
2007-07-28 14:41:46.453	2007-07-28 14:41:46.453	0
2007-07-28 14:46:47.122	2007-07-28 14:46:47.122	0
2007-07-28 14:52:48.012	2007-07-28 14:52:48.012	0
2007-07-28 14:58:48.902	2007-07-28 14:58:48.902	0
2007-07-28 15:03:49.635	2007-07-28 15:03:49.635	0
2007-07-28 15:08:50.361	2007-07-28 15:08:50.361	0
2007-07-28 15:13:51.098	2007-07-28 15:13:51.098	0
2007-07-28 15:19:51.988	2007-07-28 15:19:51.988	0
2007-07-28 15:25:52.880	2007-07-28 15:25:52.880	0
2007-07-28 15:30:53.612	2007-07-28 15:30:53.612	0
2007-07-28 15:35:54.337	2007-07-28 15:35:54.337	0
2007-07-28 15:40:55.073	2007-07-28 15:40:55.073	0

A.2. Second example

sip	dip	sp	dp	proto	flags	pkt	oct
122.166.82.184	6.71.66.160	60483	8000	6	0	1	60
122.166.82.184	6.71.66.160	60483	8000	6	2	4	240
122.166.82.184	6.71.66.160	60483	8000	6	0	1	60
122.166.82.184	6.71.66.160	60483	8000	6	0	1	60
122.166.82.184	6.71.66.160	60483	8000	6	0	1	60
122.166.82.184	6.71.66.160	56927	8000	6	0	1	60
122.166.82.184	6.71.66.160	56927	8000	6	2	4	240
122.166.82.184	6.71.66.160	56927	8000	6	0	1	60
122.166.82.184	6.71.66.160	56927	8000	6	0	1	60
122.166.82.184	6.71.66.160	56927	8000	6	0	1	60

start	end	duration
2007-07-28 14:53:36.972	2007-07-28 14:53:36.972	0
2007-07-28 14:53:37.018	2007-07-28 14:53:58.018	21
2007-07-28 14:53:39.981	2007-07-28 14:53:39.981	0
2007-07-28 14:53:45.993	2007-07-28 14:53:45.993	0
2007-07-28 14:53:57.965	2007-07-28 14:53:57.965	0
2007-07-28 15:36:18.851	2007-07-28 15:36:18.851	0
2007-07-28 15:36:18.896	2007-07-28 15:36:39.896	21
2007-07-28 15:36:21.857	2007-07-28 15:36:21.857	0
2007-07-28 15:36:27.869	2007-07-28 15:36:27.869	0
2007-07-28 15:36:39.840	2007-07-28 15:36:39.840	0

A.3. Third example

sip	dip	sp	dp	proto	flags	pkt	oct
122.166.64.100	122.166.229.157	445	3492	6	0	1	46
122.166.64.100	122.166.229.157	445	3492	6	0	1	46
122.166.64.100	122.166.229.157	445	3492	6	0	1	46
122.166.64.100	122.166.229.157	445	3492	6	0	1	46
122.166.64.100	122.166.229.157	445	3492	6	0	1	46
122.166.64.100	122.166.229.157	445	3492	6	0	1	46
122.166.64.100	122.166.229.157	445	3492	6	0	1	46
122.166.64.100	122.166.229.157	445	3492	6	0	1	46
122.166.64.100	122.166.229.157	445	3492	6	0	1	46
122.166.64.100	122.166.229.157	445	3492	6	0	1	46

start	end	duration
2007-07-28 14:46:18.510	2007-07-28 14:46:18.510	0
2007-07-28 14:52:18.509	2007-07-28 14:52:18.509	0

```
2007-07-28 14:58:18.500 2007-07-28 14:58:18.500         0
2007-07-28 15:04:18.500 2007-07-28 15:04:18.500         0
2007-07-28 15:10:18.493 2007-07-28 15:10:18.493         0
2007-07-28 15:15:18.518 2007-07-28 15:15:18.518         0
2007-07-28 15:21:18.516 2007-07-28 15:21:18.516         0
2007-07-28 15:27:18.506 2007-07-28 15:27:18.506         0
2007-07-28 15:33:18.566 2007-07-28 15:33:18.566         0
2007-07-28 15:39:18.527 2007-07-28 15:39:18.527         0
```

APPENDIX B

Example calculations

This appendix provides an example of the calculations done in Experiments 1, 2 and 3. All calculations are done on the connection given in Appendix A.1. First off, note that the connection consists of twelve flows, which satisfies the minimum requirement of ten flows. Furthermore, the duration (time between start time of last and first flow) is given by the difference between 14:41:46.453 o'clock and 15:40:55.073 o'clock, which is 59:08.620, or almost sixty minutes, (much) more than the required sixty seconds. The interarrival times of the flows in seconds are given by the following sequence of eleven values: 300.669, 360.89, 360.89, 300.733, 300.726, 300.737, 360.89, 360.892, 300.732, 300.725, 300.736.

B.1. Experiment 1

For Experiment 1, the metric is the sample standard deviation of the interarrival times Equation (5.1), which amounts to 30.4. Hence, this connection is not periodic under the metric of Experiment 1.

B.2. Experiment 2

For Experiment 2, we first compute the autocorrelation function Equation (6.2). The values of the ACF are $r_0 = 1154006$, $r_1 = 1146765$, $r_2 = 1139526$, $r_3 = 1139529$, $r_4 = 1143145$, $r_5 = 1150383$, $r_6 = 1150383$, $r_7 = 1143145$, $r_8 = 1139529$, $r_9 = 1139526$, $r_{10} = 1146765$. There is no k for which $r_{k-1} \leq r_k$ and $r_k \geq r_{k+1}$, so there are no local peaks. Hence, this connection is not periodic under the metric of Experiment 2.

To show what is done when local peaks are found, suppose that r_2, r_4 and r_6 are local peaks. This would make $p = 2$ a candidate period, since p, $2p$ and $3p$ are local peaks. By Equation (6.7), the corresponding score would be 45.3, which is too high to be periodic.

B.3. Experiment 3

For Experiment 3, Equation (7.12) is calculated for $p = 1, \ldots, 5$. The dimensions considered are the interarrival times, the duration, number of bytes and number of packets. This gives scores of 0.024 for $p = 1$, 0.035 for $p = 2$, 0.045 for $p = 3$, 0.050 for $p = 4$ and 0.00003 for $p = 5$. Hence the connection is deemed to be periodic with period 5.

Bibliography

[1] Zhenhui Li et al. 'Mining Periodic Behaviors for Moving Objects'. In: *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2010, pp. 1099–1108. DOI: 10.1145/1835804.1835942 (cit. on p. 1).

[2] Pablo Huijse et al. 'An Information Theoretic Algorithm for Finding Periodicities in Stellar Light Curves'. In: *IEEE Transactions on Signal Processing* 60.10 (2012), pp. 5135–5145. DOI: 10.1109/TSP.2012.2204260. arXiv: 1212.2398v1 (cit. on pp. 1, 8).

[3] B. Seaton. 'The Detection of Periodicity in Irregular Data'. In: *Journal of Theoretical Biology* 63.2 (1976), pp. 311–324. DOI: 10.1016/0022-5193(76)90036-9 (cit. on pp. 1, 7).

[4] C.I. Bliss. *Statistics in Biology: Statistical Methods for Research in the Natural Sciences*. Vol. 2. McGraw-Hill, 1970. Chap. 17 (cit. on pp. 1, 7, 8).

[5] Leyla Bilge et al. 'Disclosure: Detecting Botnet Command and Control Servers Through Large-Scale NetFlow Analysis'. In: *Proceedings of the 28th Annual Computer Security Applications Conference*. ACM, 2012, pp. 129–138. DOI: 10.1145/2420950.2420969 (cit. on pp. 1, 24).

[6] Xinming He et al. 'Remote Detection of Bottleneck Links Using Spectral and Statistical Methods'. In: *Computer Networks* 53.3 (2009), pp. 279–298. DOI: 10.1016/j.comnet.2008.10.001 (cit. on pp. 1, 25).

[7] Chen-Mou Cheng, H.T. Kung and Koan-Sin Tan. 'Use of Spectral Analysis in Defense Against DoS Attacks'. In: *Global Telecommunications Conference*. Vol. 3. IEEE, 2002, pp. 2143–2148. DOI: 10.1109/GLOCOM.2002.1189011 (cit. on p. 1).

[8] Rafael Ramos Regis Barbosa, Ramin Sadre and Aiko Pras. 'Towards Periodicity Based Anomaly Detection in SCADA Networks'. In: *17th Conference on Emerging Technologies & Factory Automation*. IEEE, 2012. DOI: 10.1109/ETFA.2012.6489745 (cit. on pp. 1, 26).

[9] *Meetnet Vlaamse Banken*. URL: http://www.meetnetvlaamsebanken.be/ (visited on 06/02/2015) (cit. on p. 2).

[10] *GenBank: G42670.1*. URL: http://www.ncbi.nlm.nih.gov/nuccore/4914668 (visited on 28/06/2015) (cit. on p. 3).

[11] B. Sujatha and S. Chenthur Pandian. 'A Survey on Periodicity Detection in Time Series Database'. In: *Journal of Global Research in Computer Science* 4.5 (2013), pp. 2010–2012 (cit. on p. 7).

[12] Shital P. Hatkar, S.H. Kadam and A.H. Syed. 'Analysis of Various Periodicity Detection Algorithms in Time Series Data with Design of New Algorithm'. In: *International Journal of Computer Applications Technology and Research* 3.4 (2014), pp. 229–239 (cit. on p. 7).

[13] W.F. Smyth. 'Computing Regularities in Strings: A Survey'. In: *European Journal of Combinatorics* 34.1 (2013), pp. 3–14. DOI: 10.1016/j.ejc.2012.07.010 (cit. on pp. 7, 10).

[14] Sheng Ma and J.L. Hellerstein. 'Mining Partially Periodic Event Patterns with Unknown Periods'. In: *Proceedings of the 17th International Conference on Data Engineering*. IEEE, 2001, pp. 205–214. DOI: 10.1109/ICDE.2001.914829 (cit. on pp. 7, 8).

[15] Neminath Hubballi and Deepanshu Goyal. 'FlowSummary: Summarizing Network Flows for Communication Periodicity Detection'. In: *Pattern Recognition and Machine Intelligence*. Ed. by Pradipta Maji et al. Vol. 8251. Lecture Notes in Computer Science. Springer, 2013, pp. 695–700. DOI: 10.1007/978-3-642-45062-4 (cit. on pp. 8, 23, 29).

[16] Sean McPherson and Antonio Ortega. *Analysis of Internet Measurement Systems for Optimized Anomaly Detection System Design*. 2009. arXiv: 0907.5233 (cit. on pp. 8, 25).

[17] Sean McPherson and Antonio Ortega. 'Detecting Low-rate Periodic Events in Internet Traffic Using Renewal Theory'. In: *International Conference on Acoustics, Speech and Signal Processing*. IEEE, May 2011, pp. 4336–4339. DOI: 10.1109/ICASSP.2011.5947313 (cit. on pp. 8, 25).

[18] Piotr Indyk, Nick Koudas and S. Muthukrishnan. 'Identifying Representative Trends in Massive Time Series Data Sets Using Sketches'. In: *Proceedings of 26th International Conference on Very Large Data Bases*. Morgan Kaufmann, 2000, pp. 363–372 (cit. on pp. 9, 61, 62).

[19] Michael Schaidnagel and Fritz Laux. 'Time Series Prediction with Automated Periodicity Detection'. In: *International Journal on Advances in Systems and Measurements* 6.3 (2013), pp. 394–404 (cit. on p. 9).

[20] Jiawei Han, Guozhu Dong and Yiwen Yin. 'Efficient Mining of Partial Periodic Patterns in Time Series Database'. In: *Proceedings of the 15th International Conference on Data Engineering*. IEEE, 1999, pp. 106–115. DOI: 10.1109/ICDE.1999.754913 (cit. on p. 11).

[21] Christos Berberidis et al. 'Multiple and Partial Periodicity Mining in Time Series Databases'. In: *Proceedings of the 15th European Conference on Artificial Intelligence*. Ed. by F. van Harmelen. IOS Press, 2002, pp. 370–374 (cit. on p. 11).

[22] Mohamed G. Elfeky, Walid G. Aref and Ahmed K. Elmagarmid. 'WARP: Time Warping for Periodicity Detection'. In: *Fifth IEEE International Conference on Data Mining*. IEEE, 2005, pp. 138–145. DOI: 10.1109/ICDM.2005.152 (cit. on p. 11).

[23] Faras Rasheed, Mohammed Alshalalfa and Reda Alhajj. 'Efficient Periodicity Mining in Time Series Databases Using Suffix Trees'. In: *IEEE Transactions on Knowledge and Data Engineering* 23.1 (2011), pp. 79–94. DOI: 10.1109/TKDE.2010.76 (cit. on p. 11).

[24] Jiong Yang, Wei Wang and Philip S. Yu. 'InfoMiner: Mining Surprising Periodic Patterns'. In: *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York, New York, USA: ACM, 2001, pp. 395–400. DOI: 10.1145/502512.502571 (cit. on p. 11).

[25] QinetiQ Ltd. *Command & Control: Understanding, Denying, Detecting*. Tech. rep. UK Centre for the Protection of National Infrastructure, 2014 (cit. on pp. 13, 15).

[26] Genevieve Bartlett, John Heidemann and Christos Papadopoulos. 'Low-Rate, Flow-Level Periodicity Detection'. In: *Conference on Computer Communications Workshops*. IEEE, 2011, pp. 804–809. DOI: 10.1109/INFCOMW.2011.5928922 (cit. on pp. 14, 26).

[27] Stefan Axelsson. 'The Base-Rate Fallacy and the Difficulty of Intrusion Detection'. In: *ACM Transactions on Information and System Security* 3.3 (2000), pp. 186–205. DOI: 10.1145/357830.357849 (cit. on p. 14).

[28] *APT1: Exposing One of China's Cyber Espionage Units*. Tech. rep. Mandiant, 2013 (cit. on p. 15).

[29] Peter Haag. *Nfdump*. 2013. URL: http://nfdump.sourceforge.net/ (cit. on p. 19).

[30] Anna Sperotto. 'Flow-Based Intrusion Detection'. PhD thesis. University of Twente, 2010. DOI: 10.3990/1.9789036530897 (cit. on p. 18).

[31] Rick Hofstede et al. 'Flow Monitoring Explained: From Packet Capture to Data Analysis with NetFlow and IPFIX'. In: *IEEE Communications Surveys & Tutorials* 16.4 (2014), pp. 2037–2064. DOI: 10.1109/COMST.2014.2321898 (cit. on pp. 18, 40).

[32] R.R.R. Barbosa et al. *Simpleweb/University of Twente Traffic Traces Data Repository*. Tech. rep. Centre for Telematics and Information Technology, University of Twente, 2010 (cit. on p. 18).

[33] Yong Qiao et al. 'Detecting P2P Bots by Mining the Regional Periodicity'. In: *Journal of Zhejiang University SCIENCE C* 14.9 (2013), pp. 682–700. DOI: 10.1631/jzus.C1300053 (cit. on pp. 23, 26, 51).

[34] *Tcpdump*. URL: http://www.tcpdump.org/ (cit. on p. 23).

[35] Yong Qiao et al. 'Detecting Parasite P2P Botnet in eMule-like Networks Through Quasi-Periodicity Recognition'. In: *14th International Conference*

on Information Security and Cryptology. Ed. by Howon Kim. Vol. 7259. Lecture Notes in Computer Science. Springer, 2012, pp. 127–139. DOI: 10.1007/978-3-642-31912-9_9 (cit. on p. 25).

[36] Sean Raymond McPherson. 'Event Based Measurement and Analysis of Internet Network Traffic'. PhD thesis. University of Southern California, 2011 (cit. on p. 25).

[37] Basil AsSadhan and José M.F. Moura. 'An Efficient Method to Detect Periodic Behavior in Botnet Traffic by Analyzing Control Plane Traffic'. In: *Journal of Advanced Research* 5.4 (2014), pp. 435–448. DOI: 10.1016/j.jare.2013.11.005 (cit. on p. 26).

[38] Nicholas Heard, Patrick Rubin Delanchy and Daniel Lawson. 'Filtering Automated Polling Traffic in NetFlow Data'. In: *Proceedings of Joint Intelligence and Security Informatics Conference*. IEEE, 2014 (cit. on p. 26).

[39] C.J. Bovy et al. 'Analysis of End-to-End Delay Measurements in Internet'. In: *Proceedings of Passive and Active Measurement*. 2002, pp. 26–33 (cit. on p. 40).

[40] Dao Ngoc Lam, Le Nhat Thang and Le Huu Lap. 'Defining Quantiles for Estimating IP Packet Delay Variation in NGN Core Networks'. In: *Rev Journal on Electronics and Communications* 1.3 (2011), pp. 175–182 (cit. on p. 40).

[41] M.W. Wong. *Discrete Fourier Analysis*. Birkhäuser, 2011. DOI: 10.1007/978-3-0348-0116-4 (cit. on p. 47).

[42] E. Weiszfeld. 'On the Point for Which the Sum of the Distances to n Given Points is Minimum (Translation of [46])'. Trans. by Frank Plastria. In: *Annals of Operations Research* 167.1 (2009), pp. 7–41. DOI: 10.1007/s10479-008-0352-z (cit. on pp. 64, 67).

[43] Junpei Sekino. 'n-Ellipses and the Minimum Distance Sum Problem'. In: *American Mathematical Monthly* 106.3 (1999), pp. 193–202. DOI: 10.2307/2589675 (cit. on p. 64).

[44] Volker Runde. *A Taste of Topology*. Springer, 2005. DOI: 10.1007/0-387-28387-0 (cit. on p. 64).

[45] Yehuda Vardi and Cun-Hui Zhang. 'The Multivariate L_1-Median and Associated Data Depth'. In: *Proceedings of the National Academy of Sciences of the United States of America* 97.4 (2000), pp. 1423–1426. DOI: 10.1073/pnas.97.4.1423 (cit. on p. 67).

[46] E. Weiszfeld. 'Sur le point pour lequel la somme des distances de n points donnés est minimum'. In: *Tôhoku Mathematical Journal (First Series)* 43 (1937), pp. 355–386 (cit. on p. 90).