



**Universidade do Minho**  
Escola de Engenharia  
Departamento de Informática

João Emanuel da Silva Mendes

**Evaluating the impact of traffic sampling  
in network analysis**

February 2022



**Universidade do Minho**  
Escola de Engenharia  
Departamento de Informática

João Emanuel da Silva Mendes

**Evaluating the impact of traffic sampling  
in network analysis**

Integrated Master's dissertation  
Integrated Master's in Informatics Engineering

Dissertation supervised by  
**Solange Rito Lima**  
**João Marco Cardoso da Silva**

February 2022

## AUTHOR COPYRIGHTS AND TERMS OF USAGE BY THIRD PARTIES

This is an academic work which can be utilized by third parties given that the rules and good practices internationally accepted, regarding author copyrights and related copyrights.

Therefore, the present work can be utilized according to the terms provided in the license bellow.

If the user needs permission to use the work in conditions not foreseen by the licensing indicated, the user should contact the author, through the RepositóriUM of University of Minho.

**License provided to the users of this work**



**Attribution-NonCommercial**

**CC BY-NC**

<https://creativecommons.org/licenses/by-nc/4.0/>

### STATEMENT OF INTEGRITY

I hereby declare having conducted this academic work with integrity. I confirm that I have not used plagiarism or any form of undue use of information or falsification of results along the process leading to its elaboration.

I further declare that I have fully acknowledged the Code of Ethical Conduct of the University of Minho.

Insert name

---

---

## ABSTRACT

---

The sampling of network traffic is a very effective method in order to comprehend the behaviour and flow of a network, essential to build network management tools to control Service Level Agreements (SLAs), Quality of Service (QoS), traffic engineering, and the planning of both the capacity and the safety of the network.

With the exponential rise of the amount traffic caused by the number of devices connected to the Internet growing, it gets increasingly harder and more expensive to understand the behaviour of a network through the analysis of the total volume of traffic. The use of sampling techniques, or selective analysis, which consists in the election of small number of packets in order to estimate the expected behaviour of a network, then becomes essential. Even though these techniques drastically reduce the amount of data to be analyzed, the fact that the sampling analysis tasks have to be performed in the network equipment can cause a significant impact in the performance of these equipment devices, and a reduction in the accuracy of the estimation of network state.

In this dissertation project, an evaluation of the impact of selective analysis of network traffic will be explored, at a level of performance in estimating network state, and statistical properties such as self-similarity and Long-Range Dependence (LRD) that exist in original network traffic, allowing a better understanding of the behaviour of sampled network traffic.

**Keywords:** Sampling, Quality of Service, Long-Range Dependence

---

## RESUMO

---

A análise seletiva do tráfego de rede é um método muito eficaz para a compreensão do comportamento e fluxo de uma rede, sendo essencial para apoiar ferramentas de gestão de tarefas tais como o cumprimento de contratos de serviço (*Service Level Agreements* - SLAs), o controlo da Qualidade de Serviço (QoS), a engenharia de tráfego, o planeamento de capacidade e a segurança das redes.

Neste sentido, e face ao exponencial aumento da quantidade de tráfego presente causado pelo número de dispositivos com ligação à rede ser cada vez maior, torna-se cada vez mais complicado e dispendioso o entendimento do comportamento de uma rede através da análise do volume total de tráfego. A utilização de técnicas de amostragem, ou análise seletiva, que consiste na eleição de um pequeno conjunto de pacotes de forma a tentar estimar, ou calcular, o comportamento expectável de uma rede, torna-se assim essencial. Apesar de estas técnicas reduzirem bastante o volume de dados a ser analisado, o facto de as tarefas de análise seletiva terem de ser efetuadas nos equipamentos de rede pode criar um impacto significativo no desempenho dos mesmos e uma redução de acurácia na estimação do estado da rede.

Nesta dissertação de mestrado será então feita uma avaliação do impacto da análise seletiva do tráfego de rede, a nível do desempenho na estimativa do estado da rede e a nível das propriedades estatísticas tais como a *Long-Range Dependence* (LRD) existente no tráfego original, permitindo assim entender melhor o comportamento do tráfego de rede seletivo.

**Keywords:** Análise seletiva, Qualidade de Serviço, Long-Range Dependence

---

## CONTENTS

---

1	INTRODUCTION	1
1.1	Motivation and Objectives	1
1.2	Research Methodology	2
1.3	Dissertation Layout	2
2	STATE OF ART	4
2.1	Sampling	4
2.1.1	Sampling Techniques	4
2.1.2	Traffic Flow Analysis through Sampling	6
2.1.3	Comparison of Sampling Techniques	7
2.2	Properties of Internet Traffic	11
2.2.1	Self-Similarity and Long Range Dependence	12
2.2.2	Methods for estimating LRD parameters $H$ and $\gamma$	13
2.2.3	Estimation Accuracy Evaluation	15
2.2.4	Autocorrelation function	19
3	DEVELOPMENT	21
3.1	Sampling Framework	21
3.2	Sampling Commands	23
3.3	Data Sets Used	23
3.3.1	OC48	23
3.3.2	OC192	24
3.4	Data Set Filtering	25
3.4.1	Mergecap	25
3.4.2	Wireshark	25
3.4.3	Editcap	27
3.4.4	TShark	29
3.5	Normalizing Data Sets	31
3.6	Algorithms Used	32
3.6.1	Aggregate variance	32
3.6.2	R/S	33
3.6.3	Periodogram	34
3.7	Selfis	35
4	TEST AND RESULT ANALYSIS	37
4.1	OC-48	37
4.1.1	Basic Statistics	37

4.1.2	Aggregate Variance	42
4.1.3	R/S	43
4.1.4	Periodogram	44
4.2	OC-192	46
4.2.1	Basic Statistics	46
4.2.2	Aggregate Variance	49
4.2.3	R/S	50
4.2.4	Periodogram	51
4.2.5	Autocorrelation Function	52
4.3	Summary	55
4.3.1	Data Volume	56
4.3.2	Throughput and Hurst Estimations	56
5	CONCLUSION	58
5.1	Future Work	58



---

## LIST OF FIGURES

---

Figure 1	Example of sampling techniques [1]	5
Figure 2	Framework design [1]	7
Figure 3	Volume of data [1]	9
Figure 4	Dispersion of estimated throughput - moderate workload [1]	11
Figure 5	Absolute estimation error of H attained by six methods (N=131072, mean, standard deviation and min-max excursion out of 100 samples) [2]	17
Figure 6	Absolute estimation error of H attained by six methods (N=1024, mean, standard deviation and min-max excursion out of 100 samples) [2]	18
Figure 7	Average mean $E[m_{\Delta i}]$ and standard deviation $E[\sigma_{\Delta i}]$ ( $i=0, \dots, 10$ ) of the H estimation errors attained by five methods (R(1) out of scale, average results on 100 pseudo-random sequences for each of 11 values $H_i$ ) [2]	19
Figure 8	Sampler4 Framework	22
Figure 9	Download command using the aria2c tool	22
Figure 10	Merging multiple pcap files to output oc192fulldata set.pcap	25
Figure 11	Wireshark OC192 5s time series I/O packets graph - 0.1s time frame	26
Figure 12	Wireshark OC192 5s time series I/O packets graph - 0.01s time frame	26
Figure 13	Wireshark OC192 5s time series I/O packets graph - 0.001s time frame	27
Figure 14	Formatting OC192 pcap with editcap -F	29
Figure 15	Formatting full OC192 data set pcap using -F flag to pcapng format	29
Figure 16	Tshark command 0.1s interval	30
Figure 17	Resulting IO statistics	31
Figure 18	OC192 SELFIS Hurst estimation - Aggregate variance	33
Figure 19	OC192 Matlab Hurst estimation - R/S	34
Figure 20	OC192 SELFIS Hurst estimation - Periodogram	35
Figure 21	Example of SELFIS software tool GUI	35
Figure 22	OC48 Original and sampled data sets Boxplot - 0.1s time frame	39
Figure 23	OC48 Original and sampled data sets Boxplot - 0.01s time frame	40
Figure 24	OC48 Original and sampled data sets Boxplot - 0.001s time frame	41

Figure 25	Bar chart of OC48 Aggregate Variance Hurst estimation per time frames	43
Figure 26	Bar chart of OC48 Rescaled Range Hurst estimation per time frames	44
Figure 27	Bar chart of OC48 Periodogram Hurst estimation per time frames	45
Figure 28	OC192 Original and sampled data sets Boxplot - 0.1s time frame	47
Figure 29	OC192 Original and sampled data sets Boxplot - 0.01s time frame	48
Figure 30	OC192 Original and sampled data sets Boxplot - 0.001s time frame	49
Figure 31	Bar chart of OC192 Aggregate Variance Hurst estimation per time frames	50
Figure 32	Bar chart of OC192 Rescaled Range Hurst estimation per time frames	51
Figure 33	Bar chart of OC192 Periodogram Hurst estimation per time frames	52
Figure 34	ACF graph of OC192 Original data set - 0.1s time frame	53
Figure 35	ACF graph of OC192 RandC data set - 0.1s time frame	53
Figure 36	ACF graph of OC192 SystC data set - 0.1s time frame	54
Figure 37	ACF graph of OC192 LP data set - 0.1s time frame	54
Figure 38	ACF graph of OC192 MuST data set - 0.1s time frame	55
Figure 39	ACF graph of OC192 SystT data set - 0.1s time frame	55
Figure 40	Original and sampled OC data sets volume	56

---

## LIST OF TABLES

---

Table 1	Traffic scenarios [1]	8
Table 2	Average use of computational resources [1]	10
Table 3	OC48 Basic statistics - 0.1s interval per sample	38
Table 4	OC48 Basic statistics - 0.01s interval per sample	40
Table 5	OC48 Basic statistics - 0.001s interval per sample	41
Table 6	OC48 Aggregate Variance - Hurst parameter estimation and Correlation Coefficient	42
Table 7	OC48 R/S - Hurst parameter estimation	44
Table 8	OC48 Periodogram - Hurst parameter estimation	45
Table 9	OC192 Basic statistics - 0.1s interval per sample	46
Table 10	OC192 Basic statistics - 0.01s interval per sample	47
Table 11	OC192 Basic statistics - 0.001s interval per sample	49
Table 12	OC192 Aggregate Variance - Hurst parameter estimation and Correlation Coefficient	50
Table 13	OC192 R/S - Hurst parameter estimation	51
Table 14	OC192 Periodogram - Hurst parameter estimation	52

---

## ACRONYMS

---

### A

ACF Autocorrelation function.

ARIMA Autoregressive Integrated Moving Average.

### G

GUI Graphical User Interface.

### I

IETF Internet Engineering Task Force.

ISP Internet Service Provider.

### L

LAN-PHY Local Area Network Physical Layer.

LD Logscale Diagram.

LP Linear Prediction.

LRD Long-Range Dependence.

### M

MAVAR Modified Allan Variance.

MHVAR Modified Hadamard Variance.

MRE Mean Relative Error.

MSE Mean Square Error.

MUST Multiadaptive Sampling Technique.

### O

OC Optical carrier.

**P**

PSD Power Spectrum Density.

**Q**

QoS Quality of Service.

**S**

SS Self-Similar.

SSSI Self-Similar with Stationary Increments.

SYSTC Systematic Count-based.

SYSTT Systematic Time-based.

**V**

VT Variance Time.

**W**

WAN-PHY Wide Area Network Physical Layer.

---

## INTRODUCTION

---

The importance of traffic characterization and analysis as a way of planning and managing a network is fundamental for its good performance. The association of network traffic with its corresponding applications allows the collection of very valuable information about the network, which helps the development of solutions to meet the Quality of Service (QoS) requirements of those applications. However, with the rise in the amount of data needed to be processed, the application of efficient traffic characterization and classification algorithms becomes increasingly cumbersome.

Furthermore, traffic modeling and representation is not simple, mainly with the introduction of the notion of Long-Range Dependence (LRD) in the 1990s, which means that the behavior of a time-dependent process shows statistically significant correlations across large time scales [3], unlike Poisson type processes which consider packet arrivals without a temporal notion, following an exponential distribution. The identification of these properties and the study of their impacts has become a relevant aspect in the analysis of network communications.

With the objective of simplifying network measurements, sampling techniques are implemented in strategic nodes of the network called measurement points.

Long-range dependence has been found to be present by many authors in hydrology [4, 5, 6, 7, 8, 9], economics [10, 11, 12], and in high speed networks [13, 14] [15]. The accuracy measured of the sampled data is then going to be evaluated versus the original traffic.

### 1.1 MOTIVATION AND OBJECTIVES

Over the last couple of years since the notion of LRD as a property of Internet traffic, there has been extensive amounts of study and research in order to prove that concept. However, very little focus has gone onto verifying if that property is still present when traffic sampling is applied, and how it affects the subsequent LRD estimation algorithms.

The main objective of this dissertation is focused on evaluating the impact of sampling network traffic on two levels: on the realistic estimation of the network state; on the statistical properties that network traffic presents. For such, it is necessary to identify partial objectives:

- study the different sampling techniques and the properties of network traffic;
- obtaining traffic and executing the different sampling techniques by the use of a specific sampling framework;
- comparing the different traffic sampling techniques in terms of performance and properties of sampled traffic;
- analysis and discussion of the obtained results.

## 1.2 RESEARCH METHODOLOGY

The dissertation will be divided in two phases. The first one is focused on studying the state of the art of sampling techniques and statistical analysis of network traffic.

After the knowledge acquired in the first phase, the second phase of testing will commence. The second phase is focused on the use of a framework to obtain sampled traffic and the following analysis of it, through comparisons between the different sampling techniques.

After using these techniques on network traffic, tests to measure the computational and statistical weight of each of these sampling techniques will be executed, such as statistical accuracy of the sampled traffic in terms of presence of LRD through the estimation of the Hurst parameter and network state estimation from basic statistics such as average data rate, variance and standard deviation.

That analysis will help understand which technique is better in different situations and the benefits or disadvantages in terms of network state, resources used, and also about which maintains the properties of the original traffic of a network.

## 1.3 DISSERTATION LAYOUT

This dissertation is divided into 5 chapters, covering the sampling processes, the properties of Internet traffic, the framework and the data sets used for sampling, tests and results, and finally, conclusions and future work.

- Chapter 1 - Introduction : This chapter sets the context of this dissertation, as well as the motivation and objectives to be accomplished. It also contains the dissertation layout explanation.
- Chapter 2 - State of the art: In this chapter, the foundations of sampling and its purpose are described. Furthermore, it also contains studies about Internet traffic properties mainly focused on LRD, and finally some estimators to measure such properties.

- Chapter 3 - Development : This chapter presents an explanation of the sampling framework used as well as the decisions that took place, the data sets used, their preparation and the tools used to achieve this, This chapter also contains a description of the algorithms later used for testing and results.
- Chapter 4 - Tests and results : This chapter contains all the results obtained and the corresponding discussion.
- Chapter 5 - Conclusion : This chapter contains the conclusion of the overall work as well as the potential future work as a way to further explore the theme of this dissertation.



---

## STATE OF ART

---

This chapter considers the main concepts of sampling, some of the most known sampling techniques and in which way the process differs between each of them, as well as how the sampling techniques can be divided into different components and their comparison in terms of computational weight. Other investigation points are properties of Internet traffic, mainly focused on how traffic can be viewed as a self-similar process and long range dependant, and finally some algorithms and processes to estimate the validity and existence of these properties with real time series.

### 2.1 SAMPLING

The analysis of the total volume of data of Internet traffic in order to model a network and perform accurate traffic classification becomes increasingly taxing on the performance of the network, reducing the overall throughput of such network and making it more likely to bottleneck the network equipment. A way to lighten that burden is by applying sampling techniques on the traffic and subsequent traffic analysis, classification and modeling algorithms, in a substantially smaller volume of data. However, there are different sampling techniques and algorithms, with varying rates of the amount of data collected, computational cost and the ability of accurately representing the network behaviour.

#### 2.1.1 *Sampling Techniques*

- **Systematic count-based (SystC):** Drives the packet selection through a deterministic and invariable function based on the packet position, using counters [1, 16]. As exemplified in Fig. 1 (a), every 5th packet is selected and captured by the sampling process.
- **Systematic time-based (SystT):** the process of the packet selection follows a deterministic function based on the arrival time at the measurement point [1, 16]. In this technique the sample size and the time between samples are set at the beginning and remain unchanged along the sampling process, as presented in Fig. 1 (b). As shown,

all packets arriving at the measurement point along a period of 100ms are selected for a sample, whereas all incoming packets along 200ms are ignored for measurement purposes.

- **Random count-based (RandC):** selects the starting points of the sampling intervals in accordance with a random process. As presented in Fig. 1 (c), in the n-out-of-N random approach, n elements are randomly selected out of the parent population that consists of N elements [1, 16].

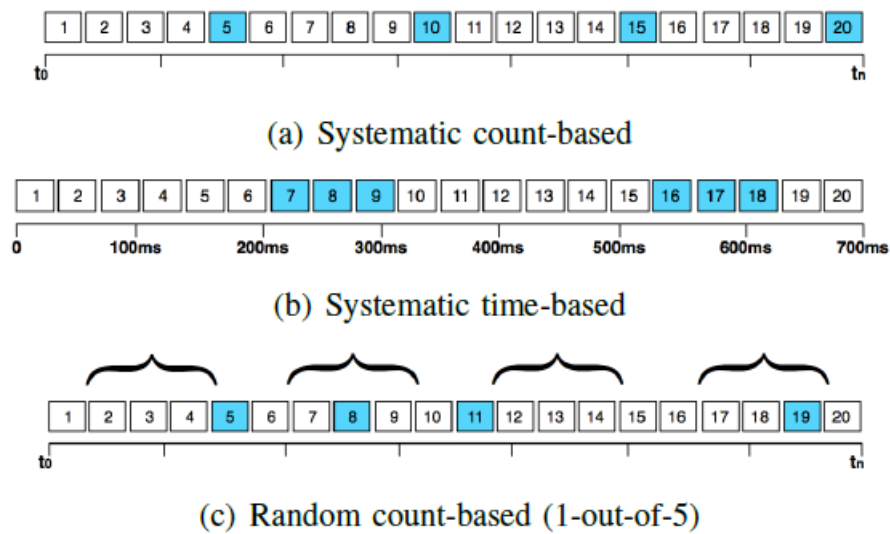


Figure 1: Example of sampling techniques [1]

- **Adaptive linear prediction:** this time-based technique uses linear prediction to identify the network activity, adjusting the sampling frequency accordingly while the sample size remains the same. This technique reduces the interval between samples when the network activity is higher than predicted. Otherwise, when less network activity than predicted is observed, the interval between samples is increased, reducing the sample frequency and, consequently, the amount of data involved in the sample process.
- **Multiadaptive sampling (MuST):** Although this technique is very similar to the adaptive linear prediction technique, the multiadaptive technique considers both the interval between samples and the sample size as adjustable parameters. Apart from increasing the sampling frequency in periods of more activity than predicted, the multiadaptive technique also reduces the sample size, avoiding the overload of the measurement point in a critical scenario of its operation. Conversely, in periods of less activity than predicted, in addition to sampling frequency reduction, the multiadaptive sampling also increases the sample size in order to acquire more information about the

network without the risk of overloading the measurement point. This technique, along with SysT achieves a better ratio of volume of data collected by computational cost.

### 2.1.2 Traffic Flow Analysis through Sampling

With the increasing importance of accurate traffic classification and characterization based on traffic sampling, the reduction of the burden of traffic analysis is no less important.

Current research on this topic is typically focused on identifying the complexity and limitations introduced by the missing data during the analysis of incomplete data traffic or traffic with gaps between data created by the sampling process.[17]

However, much of the research on this topic only consider the classic sampling techniques such as systematic count-based or random count-based approaches, which resort to a deterministic or probabilistic function, while more recent techniques such as adaptive sampling or time-based sampling are not covered.

Considering IETF PSAMP work [16, 17] and recent sampling proposals, a sampling taxonomy is used to identify the inner characteristics distinguishing sampling techniques and help defining new ones which can be adjusted to each traffic/service measurement scenario. The taxonomy defines that sampling techniques can be classified into three well-defined components according to the granularity, selection scheme and selection trigger in use. Then each component is further divided into a set of approaches commonly followed in existing sampling techniques.

1. **Granularity** - identifies the atomicity of the element under analysis in the sampling process: in flow-level approach, the sampling process is only applied to packets belonging to a specific set of flows of interest; in packet-level approach, packets are eligible as independent entities.
2. **Selection scheme** - identifies the function defining which traffic packets will be selected and collected; this scheme may follow a systematic approach, in which the process of packet selection is ruled by a deterministic function that imposes a fixed sampling frequency, independently of the packet contents or treatment; a random approach, that rules the sampling frequency through a random process, usually resorting to a pseudo-random generator or to a probabilistic function; or an adaptive approach, in which the sampling technique is endowed with the ability to change the selection of packets during the course of measurements aiming to identify the most important parts of a traffic stream according to the measurement needs or to save network resources during critical periods.
3. **Selection trigger** - determines the spatial and temporal sample boundaries; it may use a time-based approach, in which the sampling beginning and end are driven based on

the packets arrival time at the measurement point; a count-based approach, in which the sampling boundaries are defined based on the packet position in the incoming stream; or an event-based approach, in which the decision on when a sample starts and ends takes into account some particular event observed in the traffic being monitored. This event may be some value in the packet contents, the treatment of the packet at the measurement point or a more complex observation.

A framework implemented in Java using `Jpcap`, connects the sampling components in order to enable a versatile deployment of sampling techniques. This framework is designed in two planes comprising the relationship among the sampling components, as illustrated in Fig. 8, and may be applied to both online and offline measurement scenarios [1].

The sampling plane has a modular design, allowing a flexible sampling technique selection and configuration. This plane is also responsible for identifying and selecting the network interface in which the sampling will be applied.

In the network plane, traffic is collected from network interfaces by applying the sample rules defined in the sampling plane. Then the collected packets are reported to be analyzed according to network task measurement needs.

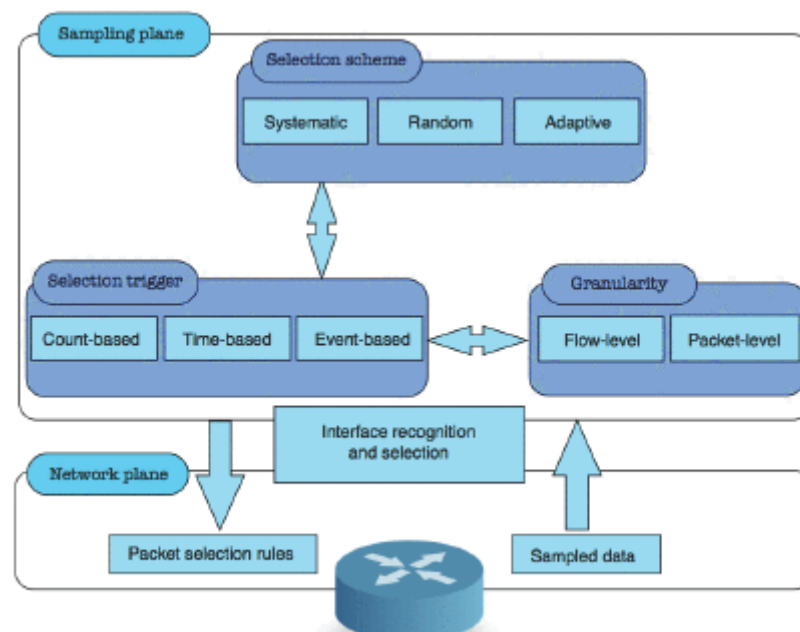


Figure 2: Framework design [1]

### 2.1.3 Comparison of Sampling Techniques

The traffic scenarios used for this analysis comparison correspond to three workload periods (low, moderate and high) in the network backbone of the University of Minho campus along

a typical workday which is then submitted to each sampling technique. The measured values are presented in the following 1.

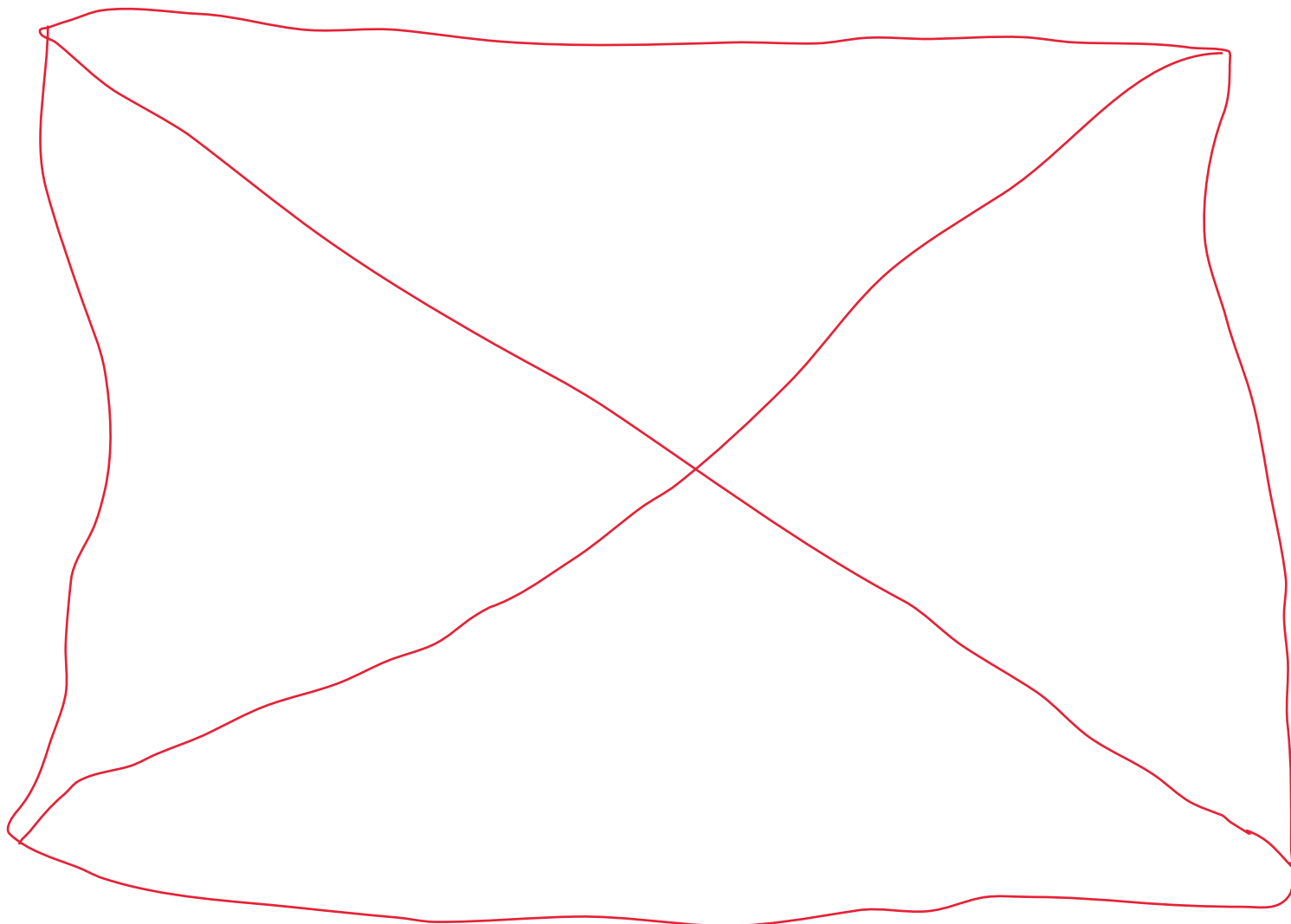
Table 1: Traffic scenarios [1]

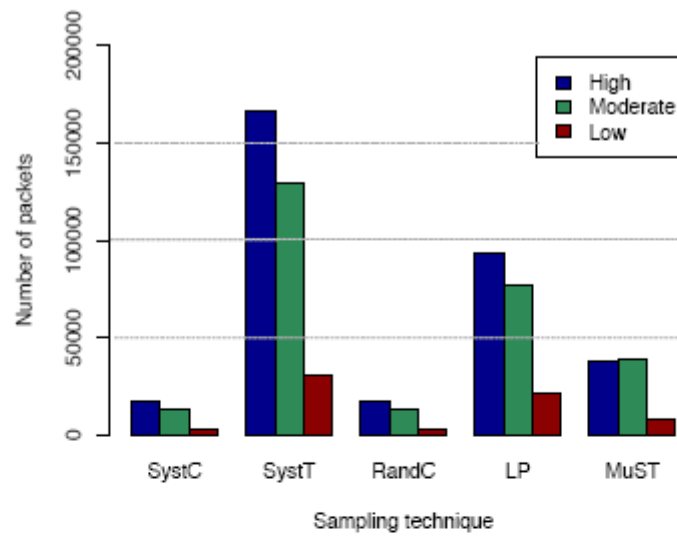
Workload scenario / Feature	Low	Moderate	High
Number of packets	311159	1273068	1718804
Volume of data (MBytes)	112.04	712.99	1063.16
Mean throughput (Mbps)	3.90	26.65	68.79
Mean packet size (Bytes)	377.58	587.26	648.59

### *Computational Weight*

The computational weight of each sampling technique is analyzed in terms of hardware resource usage (CPU Load and Memory) and volume of sampling data stored. While resource usage may impact the performance of the measurement point, data volume affects the bandwidth required by measurement data as well as the storage and processing overhead [1, 18].

Regarding the volume of data collected and stored along the sampling process, the count-based techniques (SystC and RandC) demonstrate less use of resources, as illustrated in Fig. 3. It also shows that MuST achieves the best results among the time-based techniques.





(a) Number of packets

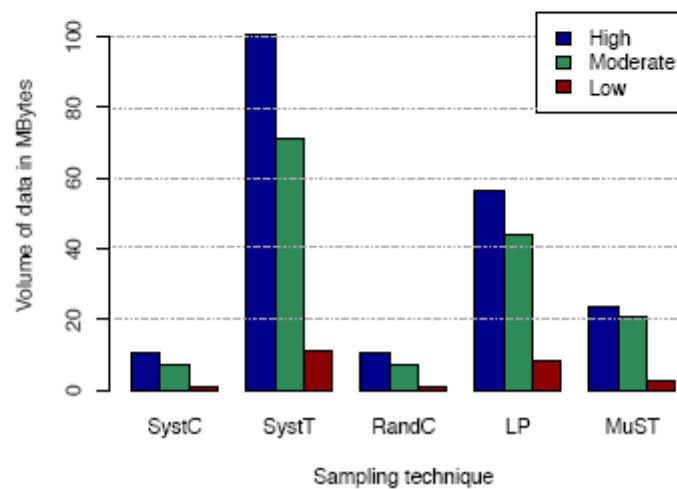


Figure 3: Volume of data [1]

Although with higher consumption of storage resources, SystT and MuST achieve a better relationship between the volume of data collected and the computational cost involved.

### Accuracy

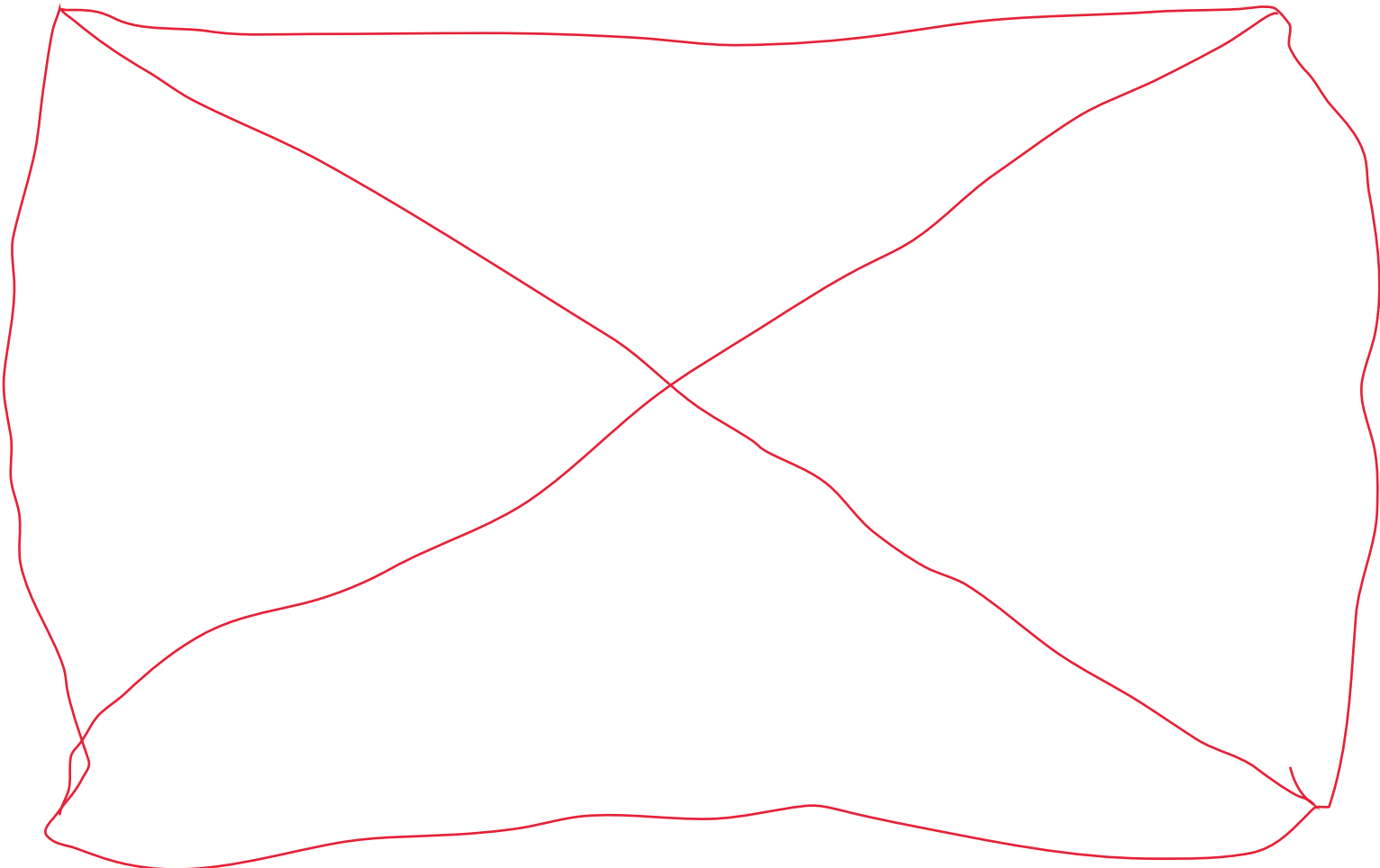
Despite the importance of reducing the usage of computational resources associated with traffic sampling, in order to be a useful tool the sampling techniques must still be able to represent the network behavior accurately [1]. A common way to achieve this goal is

Table 2: Average use of computational resources [1]

Parameter	SystC	SystT	RandC	LP	MuST
Low workload					
CPU load (%)	5.03	14.55	5.50	27.35	8.82
Memory (kBytes)	76566	95900	81222	82440	85295
Moderate workload					
CPU load (%)	10.80	17.95	16.86	96.68	10.72
Memory (kBytes)	80773	96410	84042	87698	84371
High workload					
CPU load (%)	14.92	20.12	18.26	97.27	10.76
Memory (kBytes)	81801	90754	86163	85551	80765

through estimating throughput, which consists in measuring the amount of sampled data in a time interval. Thereby, the accuracy in estimating traffic behavior is analyzed through instantaneous throughput, which is the throughput estimated in each sample along the measurement process, as well as mean throughput, i.e., the total estimated load, and its mean relative error (MRE). In addition, the accuracy in instantaneous throughput estimation is measured using the variance, where a smaller variance means a more accurate estimation. This comparison is performed calculating the mean square error (MSE), a common metric to compare estimators [1, 18].

Fig. 4 details where each point corresponds to one sample and the values closer to the reference line indicate a lower estimation error and an overall stability of its algorithms.



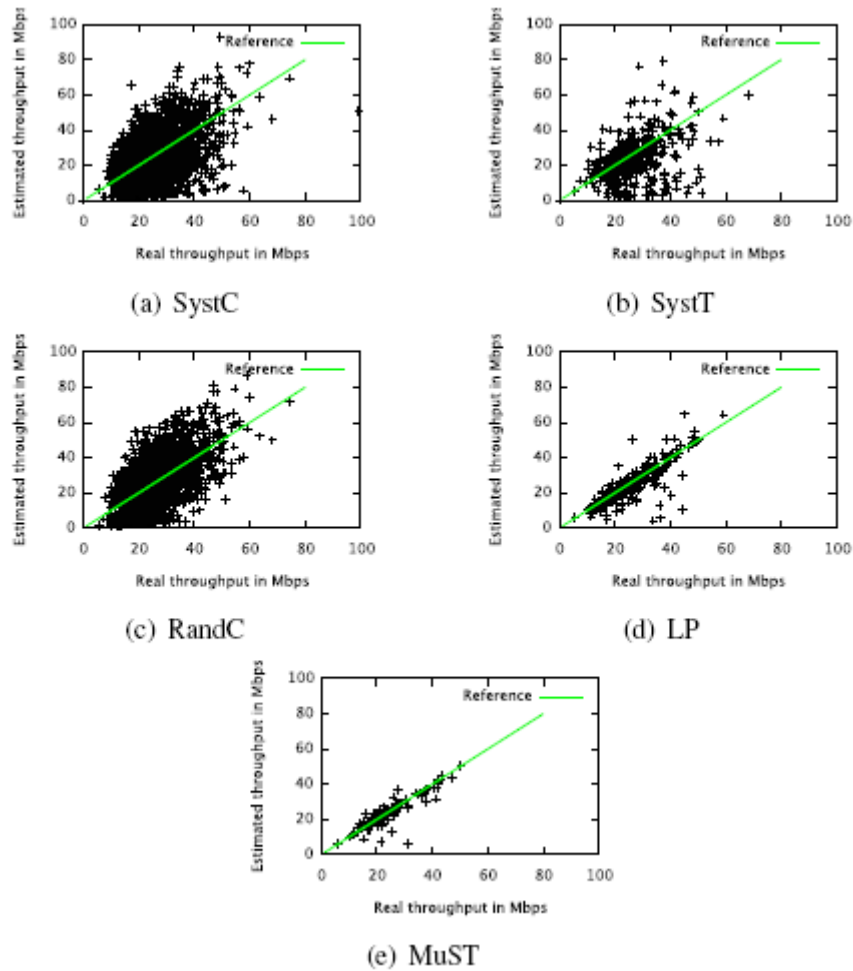


Figure 4: Dispersion of estimated throughput - moderate workload [1]

## 2.2 PROPERTIES OF INTERNET TRAFFIC

Self-similarity and scaling phenomena have dominated backbone Internet traffic analysis for the past decade. With the identification of long-range dependence (LRD) in network traffic, the research community has undergone a paradigm shift from Poisson and memory-less processes to identification of LRD and bursty processes. Despite its widespread use, though, LRD analysis is hindered by the difficulty of actually identifying dependence and estimating its parameters unambiguously [3].



### 2.2.1 Self-Similarity and Long Range Dependence

Packet traffic exhibits intriguing temporal correlation properties, such as self-similarity and long memory (long-range dependence) [2].

Interestingly, if we plot packet or byte traffic arrivals at coarser scales, say every 0.1 second, every second, or every 10 seconds, we obtain a rather unexpected result. Instead of smoother and smoother arrival counts as we would expect, we always observe a process that is almost as variable as the one observed at the finer 2 scales [19].

This property of the variance in packet or byte arrivals in Internet traffic, which is known as self-similarity or scale-invariance holds true for different time granularities or scales, from a few hundred milliseconds up to hundreds of seconds.

In a self-similar random process, a dilated portion of a realization has the same statistical characterization than the whole. “Dilating” is applied on both amplitude and time axes of the sample path, according to a scaling parameter  $H$  called Hurst parameter. On the other hand, long-range dependence (LRD) is a long-memory property observed on large time scales.

Under some hypotheses, the integral of a LRD process is self-similar with  $H$  related to  $\gamma$  (e.g., fractional Brownian motion, integral of fractional Gaussian noise).

Over the years a substantial amount of attention has been devoted to designing and perfecting algorithms for estimating parameters  $H$  and  $\gamma$  of data sequences supposed LRD.

A random process  $X(t)$  (say, cumulative packet arrivals in time interval  $[0, t]$ ) is said to be self-similar, with scaling parameter of self-similarity or Hurst parameter  $H > 0, H \in R$  if

$$X(t) =_d a^{-H} X(at) \quad (1)$$

for all  $a > 0$ , where  $=_d$  denotes equality of all finite-order distributions [2].

The class of self-similar (SS) processes is usually restricted to that of self-similar processes with stationary increments (SSSI), which are “integral” of a stationary process [2].

$H$  characterizes SS (Self-Similar) processes, but it is often used to label also the LRD increments of SSSI (Self-Similar with stationary increments) processes. For this dissertation it will follow this common custom: the expression “Hurst parameter of a LRD process” (characterized by  $\gamma$ ) denotes actually the parameter  $H = (\gamma + 1)/2$  of its integral SSSI parent process.

The predominant way to quantify LRD is through the Hurst exponent, which is a scalar [3], and can not be calculated definitively, only estimated.

There are two categories of Hurst exponent estimators: those operating in the time domain, and those operating in frequency or wavelet domain.

Despite the Poisson characteristics of packet arrivals, traces and analyses agreed with previous findings, showing that LRD characterizes backbone traffic [2].

### 2.2.2 Methods for estimating LRD parameters $H$ and $\gamma$

#### Variance-Time Plot

The basic Variance-Time (VT) plot method [2, 20, 21, 22] studies the variance of aggregated time series, calculated with samples computed by averaging non-overlapping data windows. By observing the decay of the variance plot, as a function of the window width,  $\gamma$  and  $H$  can be estimated [2].

The VT plot method [2, 20, 22] studies the variance  $\sigma^2(m)$  of the aggregated sequence obtained by dividing the LRD series  $\{x_j\}$  into non-overlapping blocks of length  $m$  and averaging them, i.e.

$$X_k^{(m)} = \frac{1}{m} \sum_{i=(k-1)m+1}^{km} x_i \quad (2)$$

for  $k=1,2,\dots,N/m$  [2]. The variance of the aggregated sequence can be estimated as the sample variance

$$\sigma^2(m) = \frac{1}{N/m} \sum_{k=1}^{N/m} [x_k^{(m)}]^2 - \left[ \frac{1}{N/m} \sum_{k=1}^{N/m} X_k^{(m)} \right]^2 \quad (3)$$

For large  $N/m$  and  $m$ , this variance obeys the power law

$$\sigma^2(m) \sim m^{\gamma-1} \sigma_x^2 \quad (4)$$

Thus, by linear regression on a log-log plot of  $\sigma^2(m)$  versus  $m$ ,  $\gamma$  and  $H$  can be estimated.

#### Rescaled Adjusted Range Statistic (R/S)

The R/S statistic is one of the main time-domain methods for LRD estimation [4, 21, 23].

Defined as

$$\frac{R(n)}{S(n)} = \frac{1}{S(n)} \left\{ \max_{0 \leq l \leq n} \left[ Y(l) - \frac{l}{n} Y(n) \right] - \min_{0 \leq l \leq n} \left[ Y(l) - \frac{l}{n} Y(n) \right] \right\} \quad (5)$$

Assuming LRD  $\{x_j\}$  such as

$$S_Y(f) \sim c_2 |f|^{-r} \text{ for } f \rightarrow 0, 0 < r < 1. \quad (6)$$

, for  $n \rightarrow \infty$  its expected value follows

$$E \left[ \frac{R(n)}{S(n)} \right] \sim C_H n^H \quad (7)$$

where  $C_R$  is a positive constant.

To estimate the Hurst parameter using R/S, the input sequence  $\{x_j\}$  is divided in K blocks. Then, for each lag- $n$ , R/S is computed at up to  $k$  starting points, taken evenly in different blocks. By linear regression on a log-log plot of  $R(n)/S(n)$  versus  $n$ ,  $H$  and  $\gamma$  can be estimated.

*Lag-1 Autocorrelation*

This time-domain method was proposed for quick identification of power-law noise [24] with integer exponent: lag-1 autocorrelation, consisting of evaluating data autocorrelation  $R(k)$  at lag  $k=1$ .

The autocorrelation of sequence  $\{x_j\}$  at lag  $k$  is estimated as

$$R(k) = \frac{\frac{1}{N} \sum_{i=1}^{N-k} (x_i - \bar{x})(x_{i+k} - \bar{x})}{\frac{1}{N} \sum_{i=1}^N (x_i - \bar{x})^2} \tag{8}$$

where  $\bar{x} = (1/N) \sum_1^N x_j$  is the mean value. The lag-1 autocorrelation is simply the value  $R(1)$  as given above. Based on this value, parameters  $\gamma$  and  $H$  can be estimated according to

$$R_Y(\delta) \sim c_1 |\delta|^{\gamma-1} \text{ for } \delta \rightarrow +\infty, 0 < \gamma < 1 \tag{9}$$

*Daubechies Wavelet Logscale Diagram*

Beyond the time and frequency domains dichotomy of the last two techniques, a breakthrough occurred when techniques based on wavelet analysis were introduced for fractional noise estimation [25, 26, 27, 28]. Due to their sensitivity to scaling phenomena over a range of scales, wavelets are well suited to detect self-similarity or other more complex scaling behaviours [2].

Assuming LRD data  $\{x_j\}$  such as 6, this method is based on observing the asymptotical power-law behaviour of the wavelet detail variances across scales

$$E[d_x(j, k)^2] \sim C2^{j\gamma} \tag{10}$$

where  $d_x(i, k)$  are the coefficients of Daubechies wavelets  $\psi_{jk}(t)$  in the decomposition of signal  $x(t)$ . These variances can be efficiently estimated as

$$\mu_j = \frac{1}{n_j} \sum_{k=1}^{n_j} d_x(j, k)^2 \tag{11}$$

where  $n_j = 2^{-j}N$  is the number of coefficients available at octave  $j$ . The log-log plot of  $\mu_j$  versus  $j$  is referred to as second-order LD. By linear regression,  $\gamma$  and  $H$  can be estimated (11).

Due to their sensitivity to scaling phenomena over a range of scales, wavelets are well suited to detect self-similarity or other more complex scaling behaviours [2].

### Modified Allan and Hadamard Variances

The Modified Allan Variance (MAVAR) is a well known time-domain quantity, purposely designed to discriminate noise types with power-law spectrum. Telecommunications standards (ANSI, ETSI, ITU-T) specify some network synchronization requirements in terms of Time Variance (TVAR), closely related to MAVAR. It was also proposed as LRD traffic analysis tool, because of its superior accuracy in estimating H and  $\gamma$ ] [2].

With a finite set of N samples  $\{x_k\}$  spaced by  $\tau_0$  over a measurement interval  $T = (N - 1)\tau_0$ , MAVAR can be estimated as

$$\text{Mod}\sigma_y^2(n\tau_0) = \frac{\sum_{j=1}^{N-3n+1} \left[ \sum_{i=j}^{i+2n-1} (x_{i+2n} - 2x_{i+n} + x_i) \right]^2}{2n^4\tau_0^2(N - 3n + 1)} \quad (12)$$

with  $n=1,2,\dots,[N/3]$ . A recursive algorithm for fast computation exists [2, 29], which cuts down the complexity of evaluating MAVAR for all  $[N/3]$  values of n to  $O(N^2)$  instead of  $O(N^3)$ .

On the other hand, MHVAR of order M can be estimated as

$$\text{Mod}\sigma_{H,M}^2(\tau) = \frac{\sum_{i=1}^{N-(M+1)n+1} \left[ \sum_{j=i}^{i+n-1} \sum_{k=0}^M \binom{M}{k} (-1)^k x_{j+kn} \right]^2}{M!n^4\tau_0^2[N - (M + 1)n + 1]} \quad (13)$$

with  $n=1,2,\dots,[N/(M+1)]$ .

Assuming LRD data  $\{x_j\}$  as 6, both MAVAR and MHVAR- $\nu_l$  are found to obey the simple power law (ideally asymptotically for  $n \rightarrow \infty, n\tau_0 = \tau$  but in practice for  $n>4$ )

$$\text{Mod}\sigma_{H,M}^2(\tau) \sim A_\mu \tau^\mu, \quad \mu = -3 + \gamma \quad (14)$$

By linear regression on log-log plot, H and  $\gamma$  can be estimated.

### 2.2.3 Estimation Accuracy Evaluation

The accuracy of the VT plot, R/S statistic, R(1), LD-3, MAVAR and MHVAR-3 methods was evaluated and compared by extensive simulations [2].

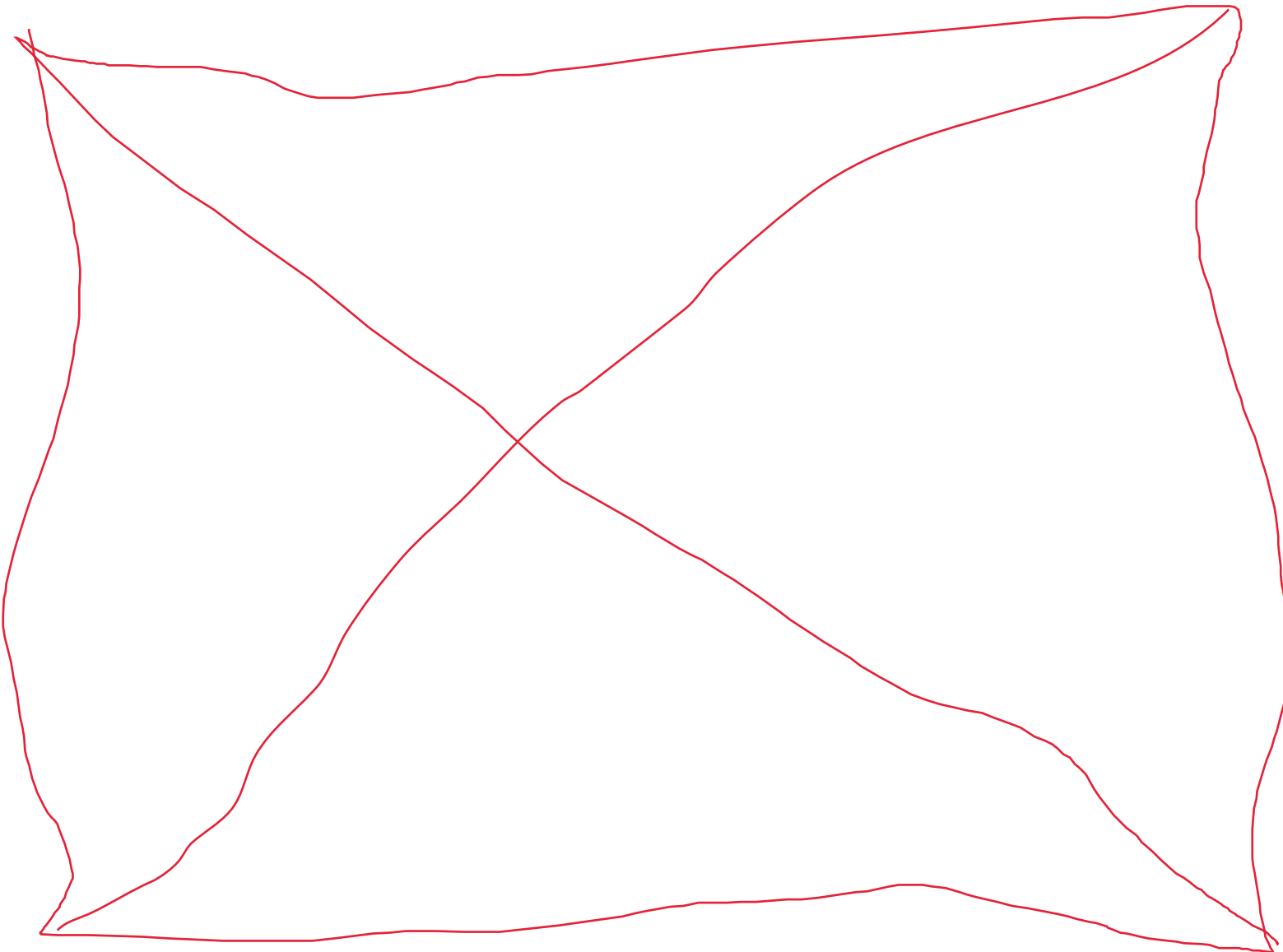
Estimation methods were applied to LRD pseudo-random data series  $\{x_k\}$  of length N generated with one-sided PSD  $S_x(f) = hf^\gamma$  ( $0 \leq \gamma \leq 1$ ) for assigned values of  $H = (1 + \gamma)/2$ . The generation algorithm is by Pason [2].

Fig. 5 compares the estimation errors  $\{\Delta_{ij}\}$  attained by the six methods on sequences of  $N=131072$  samples. For each value  $H_l$ , the mean  $m_{\Delta l}$  (dot) standard deviation  $\pm\sigma_{\Delta l}$  (thick bar) and maximum-minimum excursion (thin bar), out of 100 estimation errors, are plotted. [2].

Fig. 6 compares the estimation errors  $\{\Delta_{ij}\}$  obtained on short sequences of  $N=1024$  samples. In examining these compared plots, it should be considered that some have different scales on  $Y$  axes, due to the large difference of accuracy attained by the methods [2].

By inspection of Figs. 5, 6 and 7 MAVAR and MHVAR-3 (plotted with same scale as LD-3) provide by far the most accurate estimates [2].

- Long sequences ( $N=131072$ ). The mean of  $\sigma_{\Delta_i}$  of MHVAR-3 estimates is -22% than that of MAVAR, which in turn is -14% than that of LD-3. This confidence gain is significant, as it is computed over 1100 independent estimates. The mean of  $\sigma_{\Delta_i}$  of VT plot and R/S statistic estimates is four times wider than that of LD-3, MAVAR and MHVAR-3.
- Short sequences ( $N=1024$ ). The mean of  $\sigma_{\Delta_i}$  of MHVAR-3 estimates is -3% than that of MAVAR, which is -54% than that of LD-3. This last method, on short sequences, gives the estimates affected by highest uncertainty.



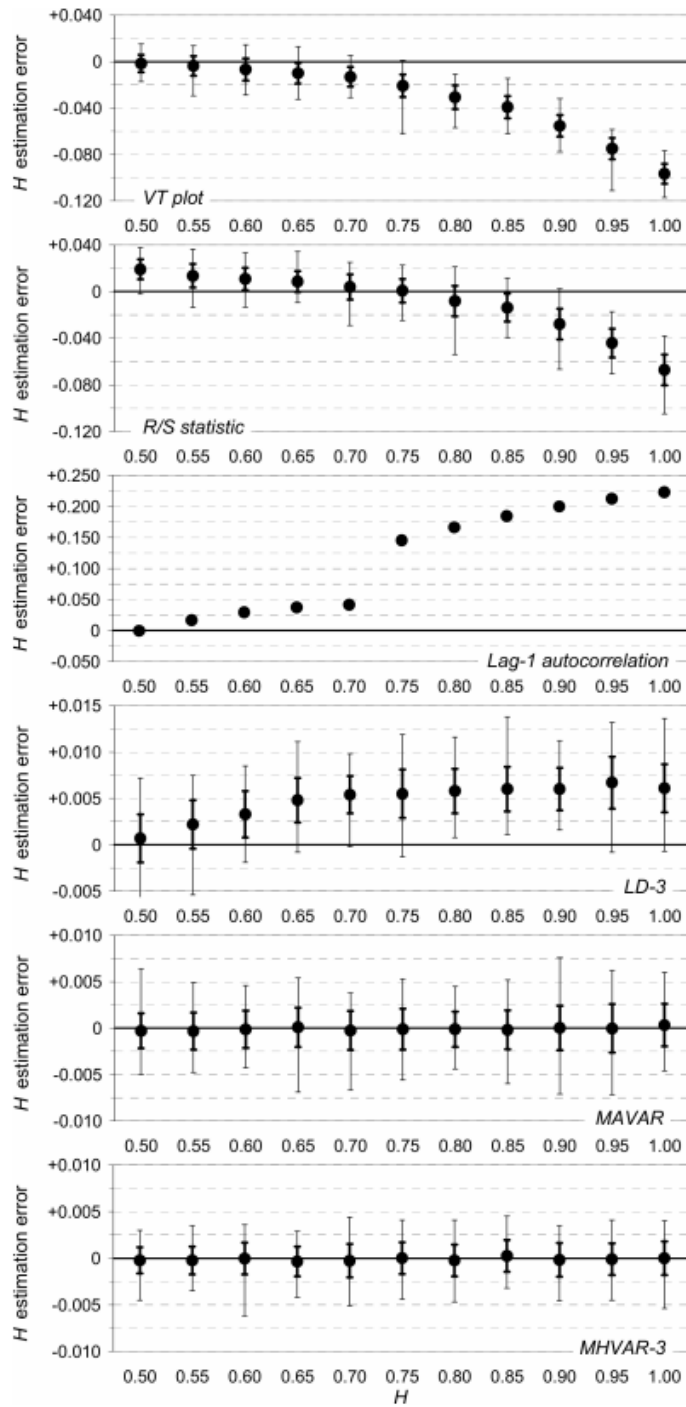


Figure 5: Absolute estimation error of H attained by six methods (N=131072, mean, standard deviation and min-max excursion out of 100 samples) [2]

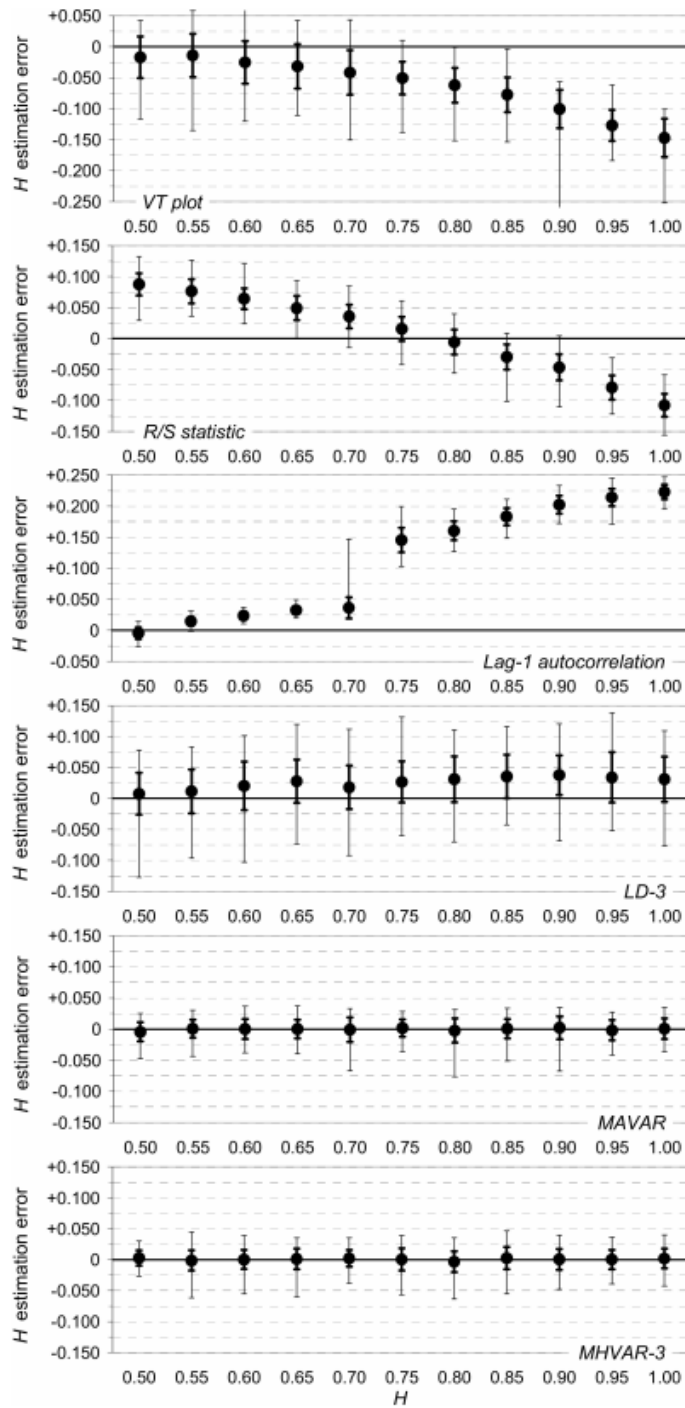


Figure 6: Absolute estimation error of H attained by six methods (N=1024, mean, standard deviation and min-max excursion out of 100 samples) [2]

MAVAR and MHVAR-3 achieve the best confidence and are not biased in H estimation. On long sequences (N=131072), the mean standard deviation of 1100 MHVAR-3 estimates

resulted 22% smaller than that of MAVAR, which in turn was 14% smaller than that of LD-3. On short sequences (N=1024), MHVAR-3 and MAVAR attained similar confidence, far better than LD-3 (mean deviation 54% smaller). This superior performance is even more significant, if we also consider the LD-3 estimation bias [17].

MAVAR and MHVAR-3 are the most accurate estimators of LRD parameters H and  $\gamma$  in terms of both confidence and bias, among all methods considered in this study. Their computational complexity is comparable to that of other methods, in particular LD, since they can be computed recursively [17, 24].

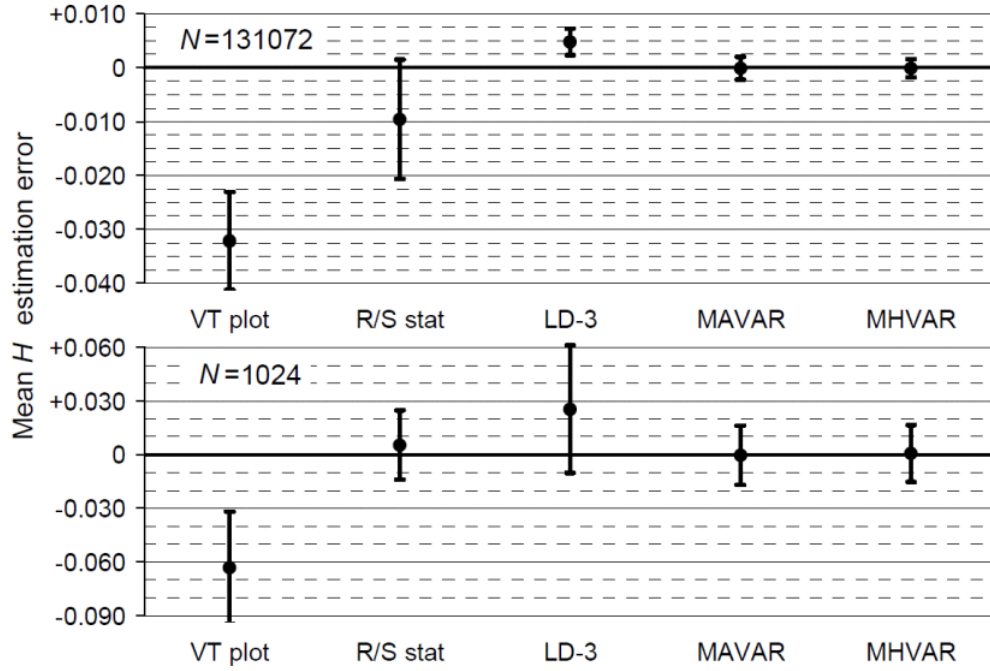


Figure 7: Average mean  $E[m_{\Delta_i}]$  and standard deviation  $E[\sigma_{\Delta_i}]$  ( $i=0, \dots, 10$ ) of the H estimation errors attained by five methods (R(1) out of scale, average results on 100 pseudo-random sequences for each of 11 values  $H_i$ ) [2]

#### 2.2.4 Autocorrelation function

Autocorrelation or ACF is a statistical measure of the relationship between a random variable and itself, depending on the lags used. As it has been previously referred, Long-range dependence measures the memory of a process. Intuitively, distant events in time are correlated. This correlation is captured by the autocorrelation function (ACF). That correlation is calculated as follows:

$$\rho(k) = \frac{E[(X_t - \mu)(X_{t+k} - \mu)]}{\sigma^2} \tag{15}$$



The autocorrelation coefficient can range between  $+1$  (very high positive correlation) and  $-1$  (very high negative correlation).

Persistent autocorrelation implies a long memory of the data-generating process or, in other words, statistical dependence between observations separated by a large number of time units [30, 31]. In contrast, if a time series has a short memory and is predictable from only its immediate past, autocorrelations decay quickly as the number of intervening observations increases [31].

If a time series shows LRD the ACF slowly decays to zero, while for short range dependence that decay occurs quickly. As explained previously, the strength of the LRD is quantified by the Hurst exponent ( $H$ ). If the number of  $H$  is between  $0.5$  and  $1$  then the series exhibits LRD and the closer the  $H$  is to  $1$ , the stronger the dependence of the process is.

---

## DEVELOPMENT

---

This chapter approaches the details of the sampling framework used in order to obtain sampled data sets, the location from where the original data sets were downloaded with a brief explanation of the network setup.

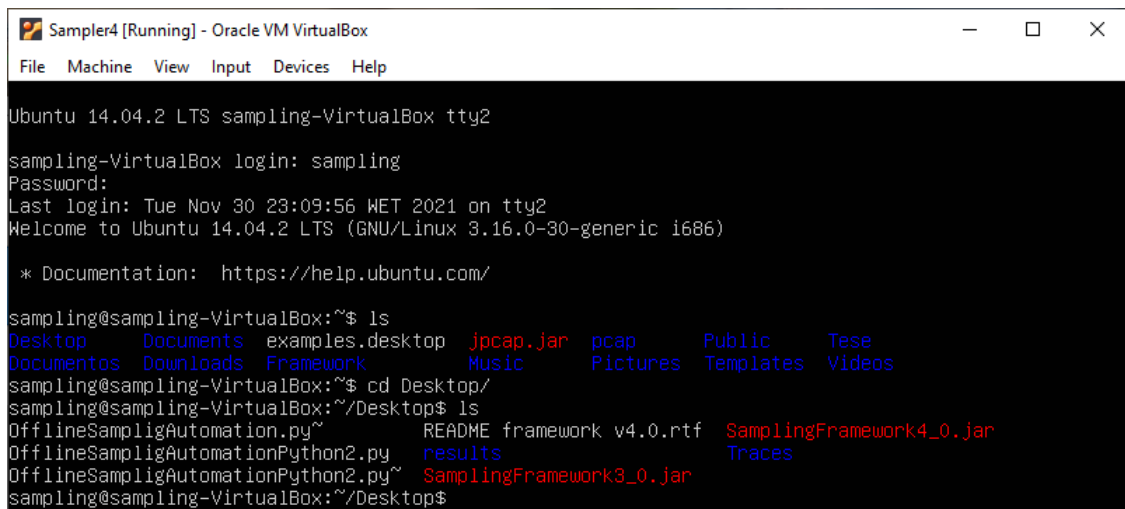
It also contains a description of all the tools used to manipulate those data sets so they can be worked with, in order to transform the packet capture files into something usable by the sampling framework and the estimation algorithms.

Furthermore, there is a description and explanation of the methods used to obtain proper inputs for the throughput and LRD estimation algorithms, referred in this chapter and later used for testing.

### 3.1 SAMPLING FRAMEWORK

This framework consists in a Virtual Machine called `Sampler4`, imported into Oracle VM VirtualBox. As it was originally developed for a 32 bit environment, the graphical display and some commands are not present and the virtual machine would just freeze and crash. In order to work around this issue you either have to work with a 32 bit machine, or turn the VM on pressing right-ctrl + F2 so the graphical display is turned off and the system turns on with the unix bash as follows in Fig. 8.

- login : sampling
- password : sampler2015



```

Sampler4 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help

Ubuntu 14.04.2 LTS sampling-VirtualBox tty2
sampling-VirtualBox login: sampling
Password:
Last login: Tue Nov 30 23:09:56 WET 2021 on tty2
Welcome to Ubuntu 14.04.2 LTS (GNU/Linux 3.16.0-30-generic i686)

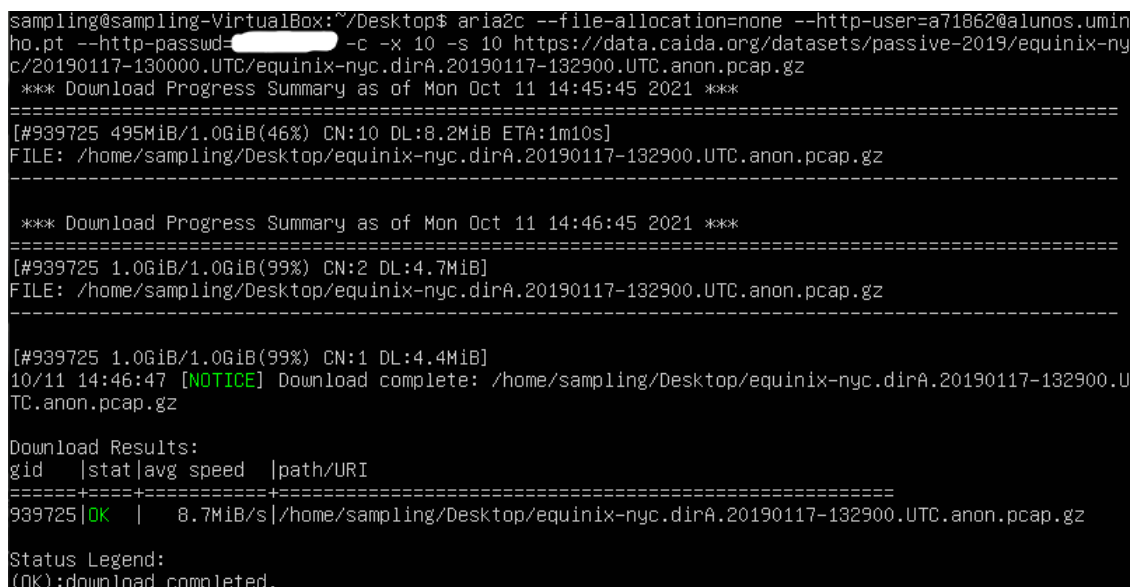
 * Documentation:  https://help.ubuntu.com/

sampling@sampling-VirtualBox:~$ ls
Desktop  Documents  examples.desktop  jpcap.jar  pcap      Public      Tese
Documentos  Downloads  Framework         Music      Pictures   Templates   Videos
sampling@sampling-VirtualBox:~$ cd Desktop/
sampling@sampling-VirtualBox:~/Desktop$ ls
OfflineSampligAutomation.py  README framework v4.0.rtf  SamplingFramework4_0.jar
OfflineSampligAutomationPython2.py  results                    Traces
OfflineSampligAutomationPython2.py  SamplingFramework3_0.jar
sampling@sampling-VirtualBox:~/Desktop$

```

Figure 8: Sampler4 Framework

The tool `aria2` was utilized to download files from `CAIDA` repository instead of more typical methods like `wget` as it is more advanced as it supports protocols such as HTTP(S), FTP, SFTP, BitTorrent, and Metalink. `aria2` can download a file from multiple sources/protocols and tries to utilize your maximum download bandwidth. It also supports downloading a file from HTTP(S)/FTP /SFTP and BitTorrent at the same time, while the data downloaded from HTTP(S)/FTP/SFTP is uploaded to the BitTorrent swarm. Using Metalink chunk checksums, `aria2` automatically validates chunks of data while downloading a file.



```

sampling@sampling-VirtualBox:~/Desktop$ aria2c --file-allocation=none --http-user=a71862@alunos.uminho.pt --http-passwd= -c -x 10 -s 10 https://data.caida.org/datasets/passive-2019/equinix-nyc/20190117-130000.UTC/equinix-nyc.dirA.20190117-132900.UTC.anon.pcap.gz
*** Download Progress Summary as of Mon Oct 11 14:45:45 2021 ***
=====
[#939725 495MiB/1.0GiB(46%) CN:10 DL:8.2MiB ETA:1m10s]
FILE: /home/sampling/Desktop/equinix-nyc.dirA.20190117-132900.UTC.anon.pcap.gz
=====

*** Download Progress Summary as of Mon Oct 11 14:46:45 2021 ***
=====
[#939725 1.0GiB/1.0GiB(99%) CN:2 DL:4.7MiB]
FILE: /home/sampling/Desktop/equinix-nyc.dirA.20190117-132900.UTC.anon.pcap.gz
=====

[#939725 1.0GiB/1.0GiB(99%) CN:1 DL:4.4MiB]
10/11 14:46:47 [NOTICE] Download complete: /home/sampling/Desktop/equinix-nyc.dirA.20190117-132900.U
TC.anon.pcap.gz

Download Results:
gid  |stat|avg speed  |path/URI
=====+=====
939725|OK  | 8.7MiB/s|/home/sampling/Desktop/equinix-nyc.dirA.20190117-132900.UTC.anon.pcap.gz

Status Legend:
(OK):download completed.

```

Figure 9: Download command using the `aria2c` tool

### 3.2 SAMPLING COMMANDS

For the sampling process itself, a Python script was used that automatically applies the different sampling techniques detailed in 2.1.1. The framework can apply these techniques to both online traffic sampling, or offline traffic in the form of pcap files which was the type of traffic used in this dissertation.

The following command applies one offline sampling technique using the SamplingFramework4\_0 taking as input a source pcap, the destination file, the sampling technique which is defined as a number (for example "SystC" corresponds to number 13), the sample size and finally the interval between samples.

For OFFLINE Sampling

```
# sudo java -jar SamplingFramework4_0.jar source.pcap Destination_file_without_
extension technique sample_size interval_between_samples
```

Another easier way of sampling, and the one used for obtaining sampled data sets was using the Python script file called *OfflineSampligAutomationPython2.py* which applies all of the different sampling techniques previously referred with reasonable parameters such as the intervals or time periods between samples to each technique.

### 3.3 DATA SETS USED

The following data sets of OC48 and OC192 were downloaded from CAIDA, following their user agreement. In the next subsections an explanation of how these network lines are capable of in terms of speeds, and how they are implemented for backbone traffic connections.

#### 3.3.1 OC48

OC-48 is a network line with transmission speeds of up to 2488.32 Mbit/s (payload: 2405.376 Mbit/s (2.405376 Gbit/s); overhead: 82.944 Mbit/s).

With relatively low interface prices, with being faster than OC-3 and OC-12 connections, and even surpassing gigabit Ethernet, OC-48 connections are used as the backbones of many regional ISPs. Interconnections between large ISPs for purposes of peering or transit are quite common. As of 2005, the only connections in widespread use that surpass OC-48 speeds are OC-192 and 10 Gigabit Ethernet.

OC-48 is also used as transmission speed for tributaries from OC-192 nodes in order to optimize card slot utilization where lower speed deployments are used. Dropping at OC-12,

OC-3 or STS-1 speeds are more commonly found on OC-48 terminals, where use of these cards on an OC-192 would not allow for full use of the available bandwidth.

The following url contains the OC48 data set used for testing.

The CAIDA UCSD Anonymized OC48 Internet Traces 2002-2003 - [2002-08-14], <https://publicdata.caida.org/datasets/passive/passive-oc48/20020814-160000.UTC/pcap/>

### 3.3.2 OC192

OC-192 is a network line with transmission speeds of up to 9953.28 Mbit/s (payload: 9510.912 Mbit/s (9.510912 Gbit/s); overhead: 442.368 Mbit/s).

A standardized variant of 10 Gigabit Ethernet (10GbE), called WAN-PHY, is designed to inter-operate with OC-192 transport equipment while the common version of 10GbE is called LAN-PHY (which is not compatible with OC-192 transport equipment in its native form). The naming is somewhat misleading, because both variants can be used on a wide area network.

In order to obtain access to the OC192 CAIDA data sets, proper authorization must be first granted, as the dataset is not public unlike the OC48 data sets. To achieve the authorization, an inquiry must first be completed with the proper reasoning for the use of the data sets wanted. After sending it, you should receive an email with the proper authentication parameters usually within a week.

The following link contains the OC192 data set used for testing. It contains anonymized passive traffic traces from CAIDA's passive monitors in 2019. It contains traffic traces from 'equinix-nyc' high-speed monitor. The CAIDA UCSD Anonymized Internet Traces - [2019-01-17], <https://www.caida.org/catalog/datasets/monitors/passive-equinix-nyc/>

#### Dataset Contents

-----

- trace files (\*.pcap.gz): compressed pcap (tcpdump) format traces
- time files (\*.times.gz): contains original nanosecond-precision timestamps

The nanosecond timestamps in each \*.times.gz line up exactly with the packets in the corresponding pcap file (containing timestamps truncated

to microsecond precision).

- stats files (\*.pcap.stats): statistics on the trace, produced by `crl_stats`

(part of the CoralReef suite of tools).

- file `md5.md5`: contains md5 checksums for all files

Traces are named using the following format:

```
{monitor}.{direction}.{start-time}.anon.pcap.gz
```

- \* monitor: equinix-nyc
- \* direction: dirA / dirB
- \* start-time: time trace began, format: yyyyymmdd-hhmmss.UTC

### 3.4 DATA SET FILTERING

In order to obtain usable data sets as input for the both the sampling framework and the algorithms to calculate basic statistics as well as the Hurst parameter, these following programs were used.

#### 3.4.1 *Mergecap*

**Mergecap** is a program that combines multiple saved capture files into a single output file specified by the `-w` argument. Mergecap knows how to read pcap and pcapng capture files, including those of tcpdump, Wireshark and other tools that write captures in those formats.

By default, Mergecap writes the capture file in pcapng format, and writes all of the packets from the input capture files to the output file.

Mergecap is able to detect, read and write the same capture files that are supported by Wireshark. The input files don't need a specific filename extension; the file format and an optional gzip compression will be automatically detected.

```
sampling@sampling-VirtualBox:~/Desktop/Traces/oc192$ mergecap *.pcap -w oc192fulldataset.pcap
```

Figure 10: Merging multiple pcap files to output oc192fulldata set.pcap

This program was used to facilitate the sampling process, as the full data set from CAIDA was split into multiple files, with the use of Mergecap, the files were combined into a single one, ordered by timestamp.

#### 3.4.2 *Wireshark*

Wireshark[32] is a GUI network protocol analyzer. It lets you interactively browse packet data from a live network or from a previously saved capture file. Wireshark's native capture file formats are pcapng format and pcap format; it can read and write both formats.. pcap format is also the format used by tcpdump and various other tools; tcpdump, when using

newer versions of the libpcap library, can also read some pcapng files, and, on newer versions of macOS, can read all pcapng files and can write them as well.

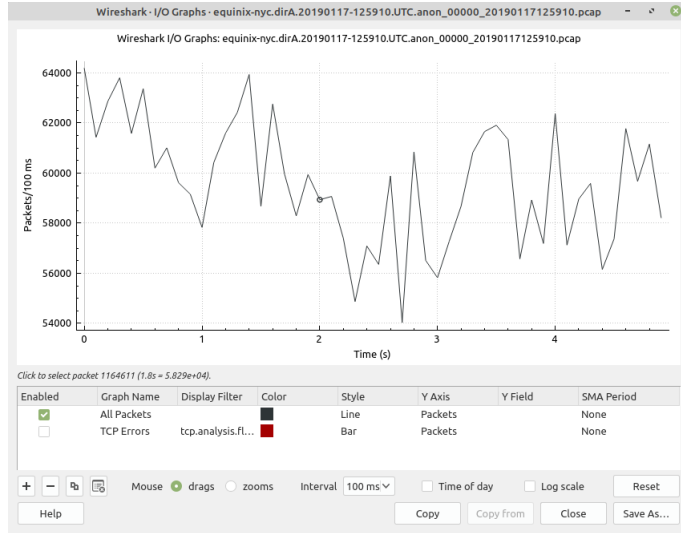


Figure 11: Wireshark OC192 5s time series I/O packets graph - 0.1s time frame

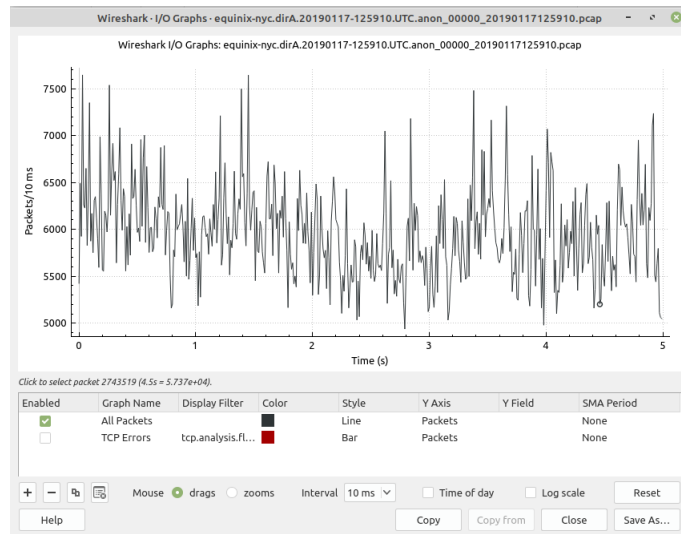
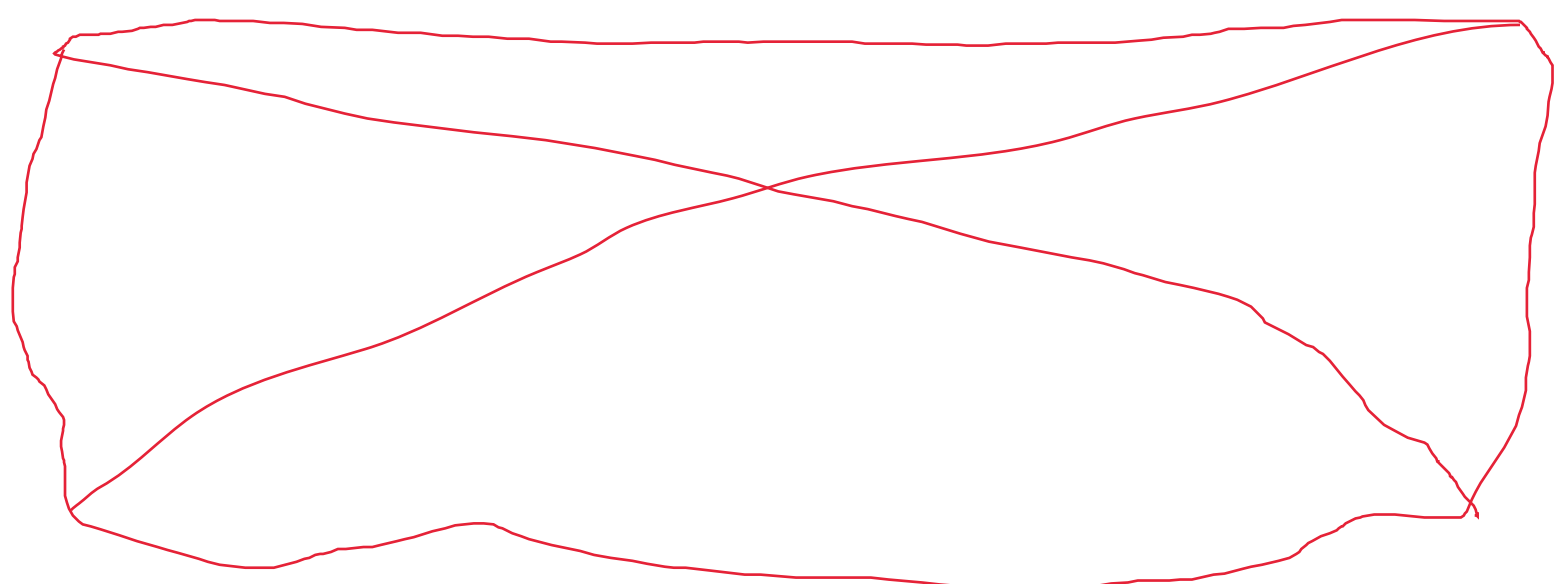


Figure 12: Wireshark OC192 5s time series I/O packets graph - 0.01s time frame



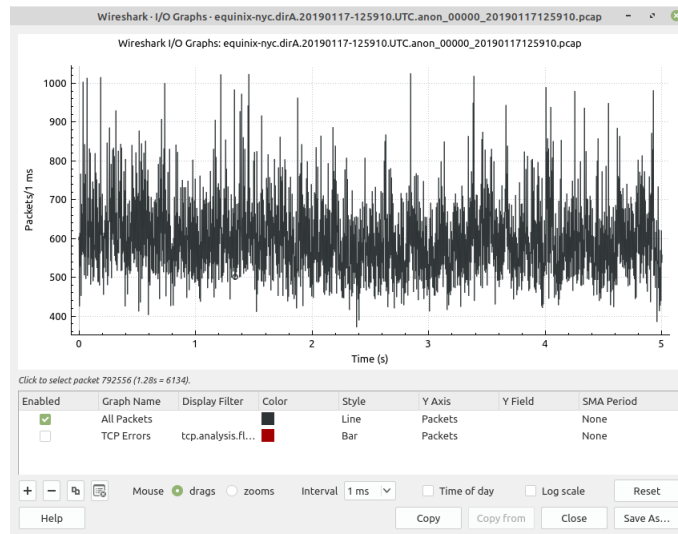


Figure 13: Wireshark OC192 5s time series I/O packets graph - 0.001s time frame

Even though Wireshark has the possibility to turn a pcap file into a CSV with all the information needed as input for the estimation algorithms which is the number of bytes for each equally spaced time interval (minimum in milliseconds), it does have its limitations considering the available number of lines in a CSV file so an alternative was instead used.

### 3.4.3 Editcap

Editcap is a program that reads some or all of the captured packets from the infile, optionally converts them in various ways and writes the resulting packets to the capture outfile (or outfiles).

By default, it reads all packets from the infile and writes them to the outfile in pcapng file format.

An optional list of packet numbers can be specified on the command tail; individual packet numbers separated by whitespace and/or ranges of packet numbers can be specified as start-end, referring to all packets from start to end. By default the selected packets with those numbers will not be written to the capture file. If the `-r` flag is specified, the whole packet selection is reversed; in that case only the selected packets will be written to the capture file.

Here are some of Editcap's functionalities:

- It can be used to remove duplicate packets. Several different options (`-d`, `-D` and `-w`) are used to control the packet window or relative time window to be used for duplicate comparison.
- Assigning comment strings to frame numbers.



- Able to detect, read and write the same capture files that are supported by Wireshark. The input file doesn't need a specific filename extension; the file format and an optional gzip compression will be automatically detected.
- Can write the file in several output formats. The -F flag can be used to specify the format in which to write the capture file; editcap -F provides a list of the available output formats.

Analysing a large pcap file cannot be done through Wireshark as the program crashes before loading the file completely. Because of this it is required to split the files that form the data set into smaller ones using Editcap. This was done through the flag -i.

-i <seconds per file> Splits the packet output to different files based on uniform time intervals using a maximum interval of <seconds per file> each. Floating point values (e.g. 0.5) are allowed.

This following Python code snippet demonstrates how Editcap was also used to split a pcap file into multiple files, as the tool Tshark (3.4.4) has a file size limit in order to work properly.

```
def edit():
    startdir='.'
    for root, dirs, files in os.walk(startdir):
        for file in files:
            if file.endswith('.pcap'):
                filename=os.path.join(root,file)
                cmd = 'editcap -i 30 "{}" "{}"'.format(filename,filename)
                os.system(cmd)
                cmd2 = 'mv {} ..'.format(filename)
                os.system(cmd2)
            return filename
```

Each output file will be created with an infix \_nnnnn[\_YYYYmmddHHMMSS] inserted before the file extension (which may be null) of outfile. The infix consists of the ordinal number of the output file, starting with 00000, followed by the timestamp of its first packet. The timestamp is omitted if the input file does not contain timestamp information.

After packets for the specified time interval are written to the output file, the next output file is opened. The default is to use a single output file. This option conflicts with -c.

The following commands correspond to the necessary formatting of the pcap files so they can be processed by the sampling framework previously described. The first one was used for the full data set of OC48 and OC192. For the OC48 data set and as the files being older, the formatting used by looking at the -F flag was through libpcap.

```
sampling@sampling-VirtualBox:~/Desktop$ editcap -F libpcap -T ether equinix-nyc.dirA.20190117-125910.UTC.anon.pcap oc192equinix125910
```

Figure 14: Formatting OC192 pcap with editcap -F

The formatting used for OC192 was pcapng as it was one that supports granularity up to nanoseconds.

```
editcap: Frame 1 of file "/media/usb/oc192fulldataset/oc192fulldataset.pcap" has a network type that differs from the network type of earlier packets, which isn't supported in a "Wireshark/tcpdump/... - nanosecond pcap" file.
sampling@sampling-VirtualBox:~/Desktop$ sudo editcap -F pcapng -T ether /media/usb/oc192fulldataset/oc192fulldataset.pcap /media/usb/oc192full.pcap
```

Figure 15: Formatting full OC192 data set pcap using -F flag to pcapng format

#### 3.4.4 TShark

TShark is a network protocol analyzer. It lets you capture packet data from a live network, or read packets from a previously saved capture file, either printing a decoded form of those packets to the standard output or writing the packets to a file. TShark's native capture file format is pcapng format, which is also the format used by Wireshark and various other tools.

Without any options set, TShark will work much like tcpdump. It will use the pcap library to capture traffic from the first available network interface and displays a summary line on the standard output for each received packet.

When run with the `-r` option, specifying a capture file from which to read, TShark will again work much like tcpdump, reading packets from the file and displaying a summary line on the standard output for each packet read. TShark is able to detect, read and write the same capture files that are supported by Wireshark. The input file doesn't need a specific filename extension; the file format and an optional gzip compression will be automatically detected.

##### *Tshark command used*

The following command was used in order to generate statistics such as packets or number of bytes from the input file.

```
-z io,stat,interval[,filter][,filter][,filter]...
```

Collect packet/bytes statistics for the capture in intervals of interval seconds. Interval can be specified either as a whole or fractional second and can be specified with microsecond (us) resolution. If interval is 0, the statistics will be calculated over all packets.

If no filter is specified the statistics will be calculated for all packets. If one or more filters are specified statistics will be calculated for all filters and presented with one column of statistics for each filter.

This option can be used multiple times on the command line.

Example: `-z io,stat,1,ip.addr==1.2.3.4` will generate 1 second statistics for all traffic to/from host 1.2.3.4.

Example: `-z "io,stat,0.001,smb\&\&ip.addr==1.2.3.4"` will generate 1ms statistics for all SMB packets to/from host 1.2.3.4.

`io,stat` can also return many more statistics and calculate `COUNT()`, `SUM()`, `MIN()`, `MAX()`, `AVG()` and `LOAD()` using a slightly different filter syntax:

Fig. 16 shows the command used to obtain the bytes per time interval.

```
plyre@VirtualBox:~/Desktop/scripts/traces$ tshark -r oc48-mfn.dirA.20020814-160000.UTC.anon_00001_20020814170020.pcap -z io,stat,"0,1"
```

Figure 16: Tshark command 0.1s interval

The following code snippet shows how the previous command was used in order to obtain the bytes per 0.1s interval, from all the pcap files in the current folder.

```
def tshark100ms():
    startdir='.'
    for root, dirs, files in os.walk(startdir):
        for file in files:
            if file.endswith('.pcap'):
                filename=os.path.join(root,file)
                packet = subprocess.run(['tshark -r "{}" -qz io,stat,0.1 >
                {}0,1.txt'.format(filename,filename)]
                ,capture_output=True, text=True, shell=True).stdout
                cmd = "sed -i '/=====/,
                $!d' {}0,1.txt".format(filename)
```

The following Fig. 17 is the text file output of one of the many text files obtained from running the previous code snippet, containing both the number of packets or frames, and bytes per each 0.1s time interval.

IO Statistics			
Duration: 20.0 secs			
Interval: 0.1 secs			
Col 1: Frames and bytes			
Interval	1	Frames	Bytes
0.0 <> 0.1		7466	3996553
0.1 <> 0.2		7543	4226120
0.2 <> 0.3		7191	4095179
0.3 <> 0.4		7650	4328132
0.4 <> 0.5		7761	4269779
0.5 <> 0.6		7643	4080955
0.6 <> 0.7		7764	4441537
0.7 <> 0.8		7542	4271268
0.8 <> 0.9		7277	4055541
0.9 <> 1.0		7620	4187687
1.0 <> 1.1		7405	4315743
1.1 <> 1.2		7535	4365146
1.2 <> 1.3		7458	4111630
1.3 <> 1.4		7864	4598860
1.4 <> 1.5		7424	4455610
1.5 <> 1.6		7692	4538904

Figure 17: Resulting IO statistics

### 3.5 NORMALIZING DATA SETS

After using all the previous tools, it is possible to extract the number of bytes per interval spread in numerous text files with extra information so the next step was sorting them to maintain the chronological order and creating a text file where each line is the number of bytes per time interval chosen previously. This file will be the input for the statistics and estimation algorithms.

The following code was used for that.

```
def iostatsSort100ms():
    for infile in sorted(glob.glob("*0,1.txt")):
        with open('{}'.format(infile)) as f:
            for line in itertools.islice(f,12,None):
                line = line[18:]
                if (line.split()[0]).isdigit():
                    ioStats01["Filename"].append(infile);
                    ioStats01["bytes"].append(int(line.split()[2]));
```

To obtain proper results the time series of Optical Carrier (OC) 48 and 192 the number of bytes were obtained for each of the following time granularities of 0.1s, 0.01s and 0.001s, therefore creating three different data sets from the original one.

Instead of using the raw number of bytes, the data was normalized, using the natural logarithmic function ( $\ln$ ), as well as the removal of all the time intervals with zero bytes from

the sampled data sets to obtain a time series more similar to the original one and without gaps of information. This is necessary as many of the previous algorithms described will give unpredictable and varying results if the removal is not done. By removing the zeros, it eliminates all the timed intervals in which the sampling techniques were not actively collecting data. The downside to this practice is the fact that a time series of network inactivity caused by failure might be inaccurately removed, which is undesirable for the network state estimations. Even so, and with the data sets used, the results should not be affected by this possibility, as moments of network outage are not present or are very brief.

The following code snippet illustrates how the final text file was obtained, containing the number of bytes per time interval after applying the `math.log` function from Python's `math` library. This file will later be used as input for the 3.7 software tool.

```
def pksTxt100ms():
    logfileBytes = open("0C192_0.1LOGBytesTXT.txt", "w")
    logZerofileBytes = open("0C192_0.1LOGZEROSBytesTXT.txt", "w")
    for infile in sorted(glob.glob("*0,1.txt")):
        with open('{}'.format(infile)) as f:
            for line in itertools.islice(f, 12, None):
                line = line[18:]
                if (line.split()[0]).isdigit():
                    if int(line.split()[2]) != 0 :
                        logfileBytes.write(str(math.log(
                            int(line.split()[2])))+"\n")
                        logZerofileBytes.write(str(math.log(
                            int(line.split()[2])))+"\n")
                    else :
                        logZerofileBytes.write(line.split()[0)+"\n")
```

### 3.6 ALGORITHMS USED

The following algorithms and the explanation of how they are processed is inspired in the following article [21]. The description of those algorithms is detailed in the next subsections.

#### 3.6.1 Aggregate variance

Divide the input time series  $X = \{X_i, i \geq 1\}$  into  $m$  blocks of size  $m$  and average within each block, that is considered the aggregated series for successive values of  $m$ . The index  $k$ , labels the block. Then take the sample variance of  $X^{(m)}(k), k = 1, 2, \dots$  of each block.

This sample variance is an estimator of  $VarX^{(m)}$ . Since, for fractional Gaussian noise and fractional ARIMA,  $VarX(m) \sim \sigma_0^2 m^\beta$  as  $m \rightarrow \infty$  where  $\beta = 2H - 2 < 0$ , it is possible to obtain an estimate for  $\beta$ , or  $H$ , by proceeding as follows.

For a given  $m$ , divide the data,  $X_1, \dots, X_N$ , into  $N/m$  blocks of size  $m$ , calculate  $X^{(m)}(k)$ , for  $k = 1, 2, \dots, N/m$ , and its sample variance

$$\widehat{Var}X^{(m)} = \frac{1}{N/m} \sum_{k=1}^{N/m} (X^{(m)}(k))^2 - \left( \frac{1}{N/m} \sum_{k=1}^{N/m} X^{(m)}(k) \right)^2 \tag{16}$$

Apply this process for different values of  $m$  and plot the logarithm of the sample variance versus  $\log m$ . Choose values of  $m$ ,  $\{m_i, i \geq 1\}$ , that are the same distance on a log scale, so that  $m_{i+1}/m_i = C$ , where  $C$  is a constant which depends on the length of the series and the desired number of points. Since  $\widehat{Var}X^{(m)}$  is an estimate of  $VarX^{(m)}$ , the resulting points should form a straight line with slope  $\beta = 2H - 2$ ,  $-1 \leq \beta < 0$ . In practice, the slope is estimated by fitting a line to the points obtained from the plot. It is assumed here that both  $m$  and  $N$  are large, and that  $m \ll N$ , so that both the length of each block, and the number of blocks is large. If  $X$  has (short-range or) no dependence, the slope obtained should equal  $-1$ .

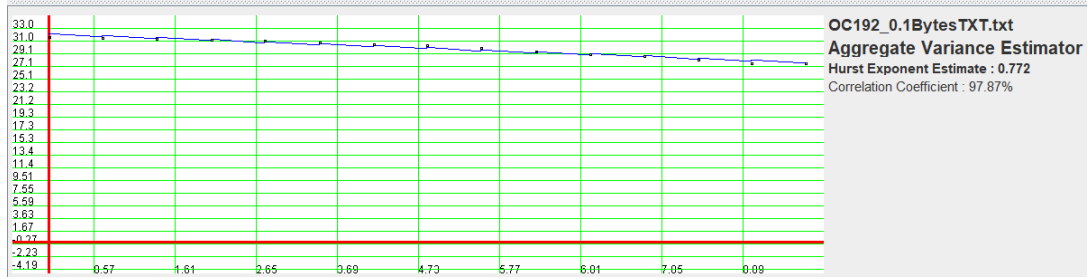


Figure 18: OC192 SELFIS Hurst estimation - Aggregate variance

### 3.6.2 R/S

This is one of the better known methods as previously referred in 2.2.2. For a time series  $X = \{X_i, i \geq 1\}$ , with partial sum  $Y(n) = \sum_{i=1}^n X_i$ , and sample variance  $S_2(n) := (1/n) \sum_{i=1}^n X_i X_i^2 - (1/n)_2 Y(n)_2$  the R/S statistic, or the rescaled adjusted range is given by :

$$\frac{R}{S}(n) := \frac{1}{S(n)} \left[ \max_{0 \leq t \leq n} \left( Y(t) - \frac{t}{n} Y(n) \right) - \min_{0 \leq t \leq n} \left( Y(t) - \frac{t}{n} Y(n) \right) \right] \tag{17}$$

For fractional Gaussian noise or fractional ARIMA,

$$E[R/S(n)] \sim C_H n^H \tag{18}$$

as  $n \rightarrow \infty$ , where  $C_H$  is another positive, finite constant not dependent on  $n$ . To determine  $H$  using the  $R/S$  statistic, proceed as follows. For a time series of length  $N$ , subdivide the series into  $K$  blocks, each of size  $N/K$ . Then, for each lag  $n$ , compute  $R(k_i, n)/S(k_i, n)$ , starting at points  $k_i = iN/K + 1, i = 1, 2, \dots$ , such that  $k_i + n \leq N$ . For values of  $n$  smaller than  $N/K$ , one gets  $K$  different estimates of  $R(n)/S(n)$ . For values of  $n$  approaching  $N$ , one gets fewer values, as few as 1 when  $n \geq N - N/K$ . Choosing logarithmically spaced values of  $n$ , plot  $\log[R(k_i, n) = S(k_i, n)]$  versus  $\log n$  and get, for each  $n$ , several points on the plot. This plot is sometimes called the *pox* plot for the  $R/S$  statistic. The parameter  $H$  can be estimated by fitting a line to the points in the *pox* plot. Since any short-range dependence in the series typically results in a transient zone at the low end of the plot do not use the low end of the plot for the purposes of estimating  $H$ . Usually, the very high end of the plot is not used as well, because there are too few points on the plot at the high end to make reliable estimates. The values of  $n$  that lie between the lower and higher cut-off points are used to estimate  $H$ .

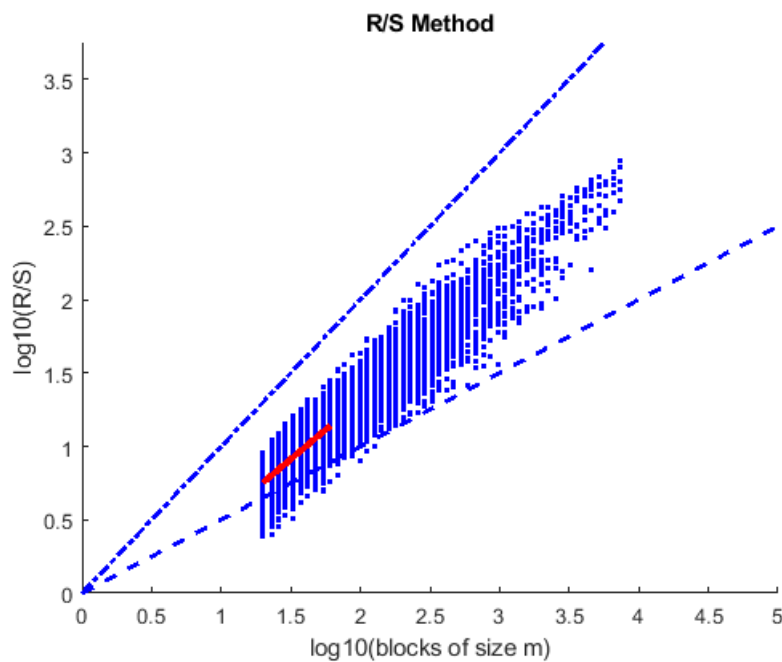


Figure 19: OC192 Matlab Hurst estimation - R/S

### 3.6.3 Periodogram

One first calculates

$$I(\lambda) = \frac{1}{2\pi N} \left| \sum_{j=1}^N X_j e^{ij\lambda} \right|^2 \tag{19}$$

where  $\lambda$  is a frequency,  $N$  is the number of terms in the series, and  $X_j$  is the data. Because  $I(\lambda)$  is an estimator of the spectral density, a series with long-range dependence should have a periodogram which is proportional to  $|\lambda|^{1-2H}$  close to the origin. Therefore, a regression of the logarithm of the periodogram on the logarithm of the frequency  $\lambda$  should give a coefficient of  $1 - 2H$ . The slope provides an estimate of  $H$ .

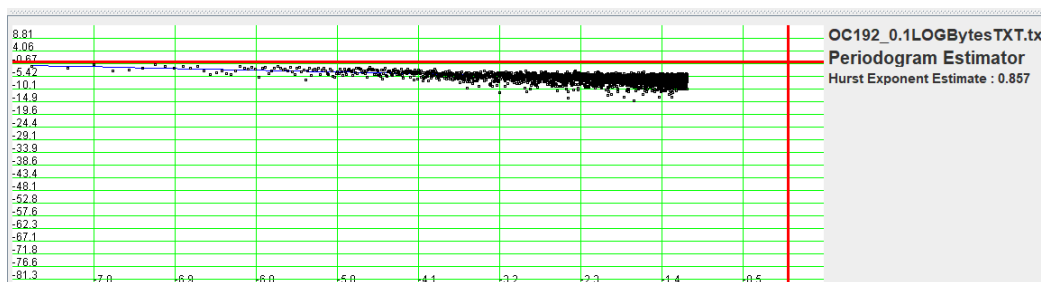


Figure 20: OC192 SELFIS Hurst estimation - Periodogram

### 3.7 SELFIS

The testing and results were obtained using the self-similarity analysis software tool SELFIS[33], capable of applying all of the Hurst estimation algorithms referred in 3.6 to an input file which contains values of a time series, It can also obtain basic statistics, the graphical display of the power spectrum function and the autocorrelation function (ACF).

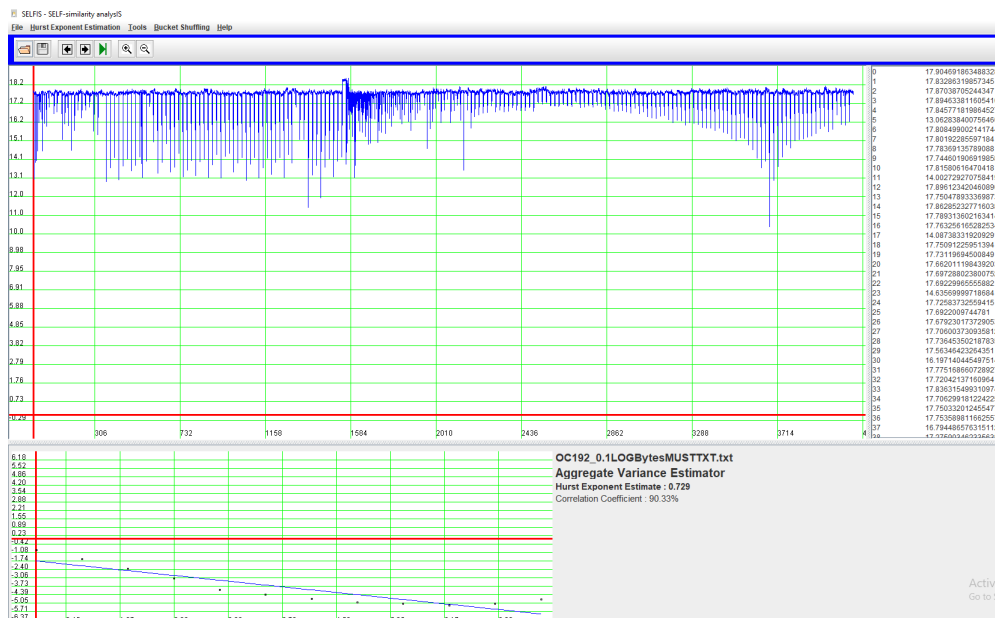


Figure 21: Example of SELFIS software tool GUI



Due to limitations in the amount of data the SELFIS software tool could import, the comparisons between the different time granularities might not be completely accurate but a close approximation, as each data set could only be partially loaded. Note that comparisons between original and sampled data sets should be accurate as they contain the same amount of values, with the exception of some sampling techniques which shortened the sampled time series data set.

The SELFIS software tool also returns the correlation coefficient in a percentage value for the Aggregate Variance. Some other estimators are also available such as the Absolute Moment, Whittle estimator, Variance of Residuals and Abry-Veitch estimator, which were not used for testing.

---

## TEST AND RESULT ANALYSIS

---

The following sections are the results obtained using the SELFIS software tool for the Hurst parameter estimations with the Aggregate Variance, R/S and Periodogram algorithms as well as the basic statistics for containing the mean throughput in the natural logarithm of the number of bytes and the conversion to Mbps. For the R/S algorithm, Matlab was used as SELFIS presented unexpected and inconsistent results.

The time frames chosen to represent the OC48 and OC192 and the respective sampled data sets were the 0.1s, 0.01s and 0.001s.

### 4.1 OC-48

With a 0.1s time frame only  $N = 65536$  samples could be imported to the SELFIS tool (out of 108000), for 0.01s  $N = 1048576$  (out of 1080000) and finally for the 0.001s time frame  $N = 8388608$  (out of 10800000) for the original data sets. Note that the OC48 data set should be 10800s long which corresponds to three hours of network traffic data. Each of the sampled data sets have a different number of samples depending on the sampling technique used.

#### 4.1.1 Basic Statistics

As it has been previously referred in 2.1.3, the accuracy in estimating traffic behavior is analyzed through instantaneous throughput, which is the throughput estimated in each sample along the measurement process, as well as mean throughput which is the total estimated load. The accuracy in instantaneous throughput estimation is measured using the variance, where a smaller variance means a more accurate estimation.

The boxplots were obtained with Matlab[34] using the following code as example.

```
Input = {OC48_0_1LOGBytes, OC48_0_1LOGBytesLP, OC48_0_1LOGBytesMuST,  
OC48_0_1LOGBytesRANDC, OC48_0_1LOGBytesSYSTC1_8,  
OC48_0_1LOGBytesSYSTC1_100, OC48_0_1LOGBytesSYSTC1_256,  
OC48_0_1LOGBytesSYSTT100_500, OC48_0_1LOGBytesSYSTT200_500,
```

```

OC48_0_1LOGBytesSYSTT500_1500, OC48_0_1LOGBytesSYSTT500_3500};

% Pad each vector with NaN values to equate lengths
maxNumEl = max(cellfun(@numel,Input));
Cpad = cellfun(@(x){padarray(x(:),[maxNumEl-numel(x),0],NaN,'post')}, Input);
% Convert cell array to matrix and run boxplot
Cmat = cell2mat(Cpad);

boxplot(Cmat,Dataset);
title('OC48 Boxplot - 0.1s time frame')
xlabel('Data set')
ylabel('Number of bytes per sample(in ln)')

```

#### *0.1s time frame data set*

On Table 3 are presented statistics such as mean (natural logarithm of the number of bytes at the 0.1s time frame interval and corresponding throughput in Megabits per second ( $e^{15.30} * 8 = 353Mbps$ ), variance, standard deviation and skewness.

The sampling techniques data set that more closely resembles the original data set in terms of resulting statistics are MuST and the count-based techniques. The SystT sampling technique presented poor results with relatively close mean average to the original but with high variance and deviation as shown in the boxplot represented in Fig. 22.

Also as expected in the case of the count-based techniques, if you multiply the interval of the sampling frequency the with the estimated throughput, that number comes close to the original data set calculated throughput, so for example SystC(1/8) :  $39 * 8 = 312Mbps$ , for SystC(1/100) :  $3.5 * 100 = 350$ , and so forth.

Table 3: OC48 Basic statistics - 0.1s interval per sample

	Mean	Throughput	Variance	Std.Deviation	Skewness
Original	15.30	353Mbps	0.006	0.079	-4.554
LP(100/200)	14.15	112Mbps	1.747	1.321	-1.649
MuST(200/500)	15.15	304Mbps	0.620	0.787	-4.699
RandC(1/100)	10.69	3.5(350)Mbps	0.021	0.145	-1.059
SystC(1/8)	13.10	39(312)Mbps	0.007	0.086	-3.569
SystC(1/100)	10.68	3.5(350)Mbps	0.021	0.146	-1.046
SystC(1/256)	9.737	1.4(358)Mbps	0.048	0.219	-0.840
SystT(100/500)	10.62	3.3Mbps	23.46	4.844	-0.158
SystT(200/500)	12.17	15.4Mbps	20.72	4.551	-0.832
SystT(500/1500)	13.78	77.2Mbps	12.44	3.527	-1.967
SystT(500/3500)	13.81	79.6Mbps	12.05	3.471	-1.985

Looking at Fig. 22, it is easy to come to the conclusion that the time-based techniques show very high variances except for MuST. The variance of the count-based techniques is very close to that of the original data set, but with a lower amount of bytes per sample, because of the packet selection through the sampling process (for the SystC(1/8) only 1 packet will be selected out of 8 making the throughput estimations lower).

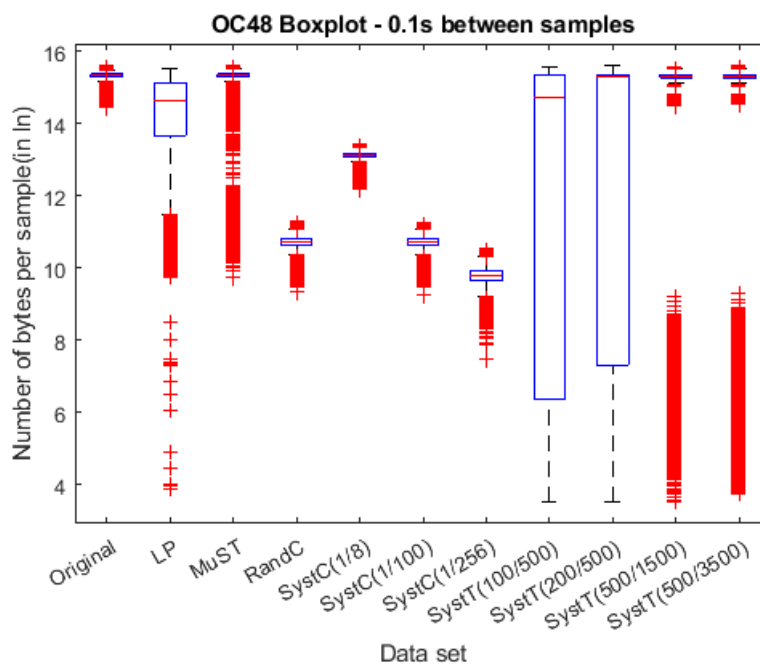


Figure 22: OC48 Original and sampled data sets Boxplot - 0.1s time frame

#### *0.01s time frame data set*

Looking at the following Table 4 of the OC48 data sets obtained basic statistics for the 0.01s time frame. At finer granularities the time-based techniques start performing better in terms of estimating the mean throughput when compared to 3. Still, SystC(1/8) and MuST are the best two sampling techniques if you want to accurately estimate throughput, as the variance observed in these data sets is very comparable to the original data set.

Looking at Fig. 23, the time-based techniques start performing better, contrary to the count-based techniques. However the SystT sampling technique still presents very high variance.

Table 4: OC48 Basic statistics - 0.01s interval per sample

	Mean	Throughput	Variance	Std.Deviation	Skewness
Original	13.02	361Mbps	0.014	0.119	-0.968
LP(100/200)	12.83	299Mbps	0.356	0.597	-6.237
MuST(200/500)	12.99	350Mbps	0.030	0.175	-6.359
RandC(1/100)	8.327	3.3(330)Mbps	0.236	0.486	-1.327
SystC(1/8)	10.82	40(320)Mbps	0.026	0.162	-0.639
SystC(1/100)	8.319	3.3(330)Mbps	0.233	0.483	-1.317
SystC(1/256)	7.164	1.0(256)Mbps	0.909	0.953	-1.152
SystT(100/500)	12.36	187Mbps	4.384	2.094	-3.107
SystT(200/500)	12.68	257Mbps	2.452	1.566	-4.538
SystT(500/1500)	12.89	317Mbps	1.005	1.002	-7.394
SystT(500/3500)	12.89	317Mbps	0.959	0.979	-7.438

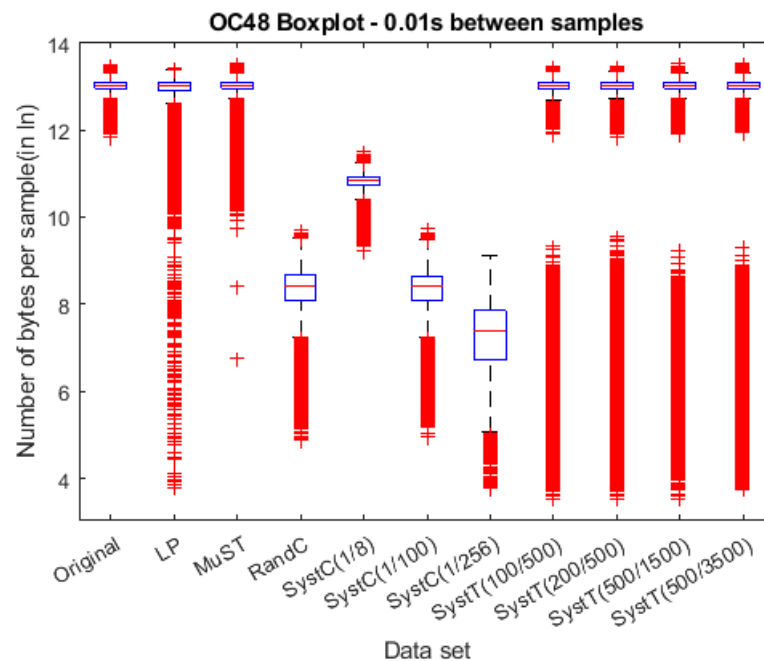


Figure 23: OC48 Original and sampled data sets Boxplot - 0.01s time frame

#### 0.001s time frame data set

Once again, looking at Table 5 presenting the resulting OC48 basic statistics for the 0.001s time frame, and comparing the original data set to the sampled ones, the time-based sampling techniques (LP, MuST and SystT) even though showing a more skewed data series, start performing better than the count-based ones for calculating throughput.

Count-based techniques show increasingly worse results with the finer the time granularity data sets, but SystC(1/8) still presented good results.

The statistics obtained from the SystT technique data set are also a lot closer relative to the original data set in terms of mean when compared to the 0.1s and 0.01s time frames, however still presenting strong skewness.

Table 5: OC48 Basic statistics - 0.001s interval per sample

	Mean	Throughput	Variance	Std.Deviation	Skewness
Original	10.69	351Mbps	0.066	0.258	-0.510
LP(100/200)	10.65	338Mbps	0.149	0.386	-7.572
MuST(200/500)	10.67	344Mbps	0.066	0.257	-1.267
RandC(1/100)	5.592	2.1(210)Mbps	2.274	1.508	-0.011
SystC(1/8)	8.420	36(288)Mbps	0.261	0.511	-1.282
SystC(1/100)	5.464	1.9(190)Mbps	2.245	1.498	0.093
SystC(1/256)	5.448	1.9(486)Mbps	2.234	1.494	-0.118
SystT(100/500)	10.63	331Mbps	0.307	0.554	-8.122
SystT(200/500)	10.65	338Mbps	0.193	0.440	-8.539
SystT(500/1500)	10.68	348Mbps	0.116	0.340	-7.355
SystT(500/3500)	10.68	348Mbps	0.113	0.337	-7.122

Looking at Fig. 24, the results are further amplified from the previous time frame plotted by Fig. 5. At this time frame, the time-based techniques are the best performers, with the count-based sampling techniques becoming less accurate than in the previous time frames.

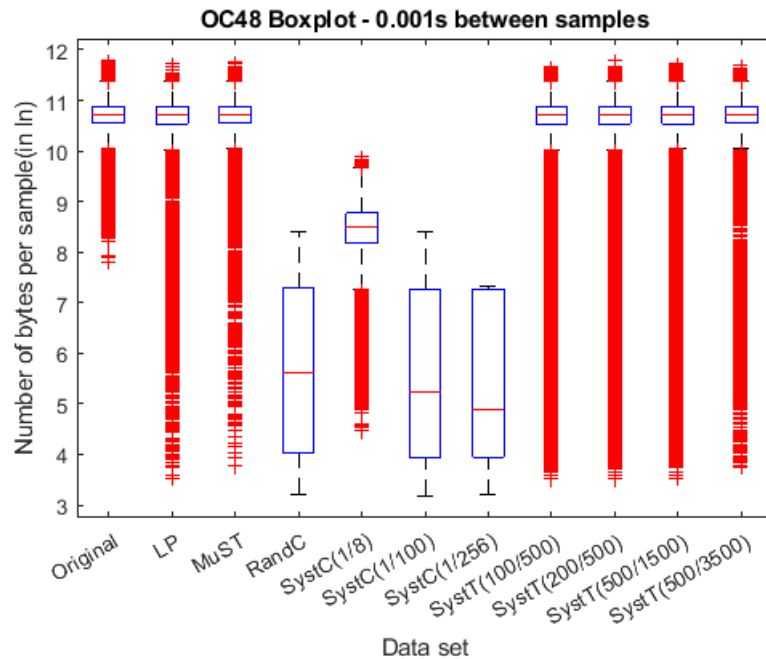


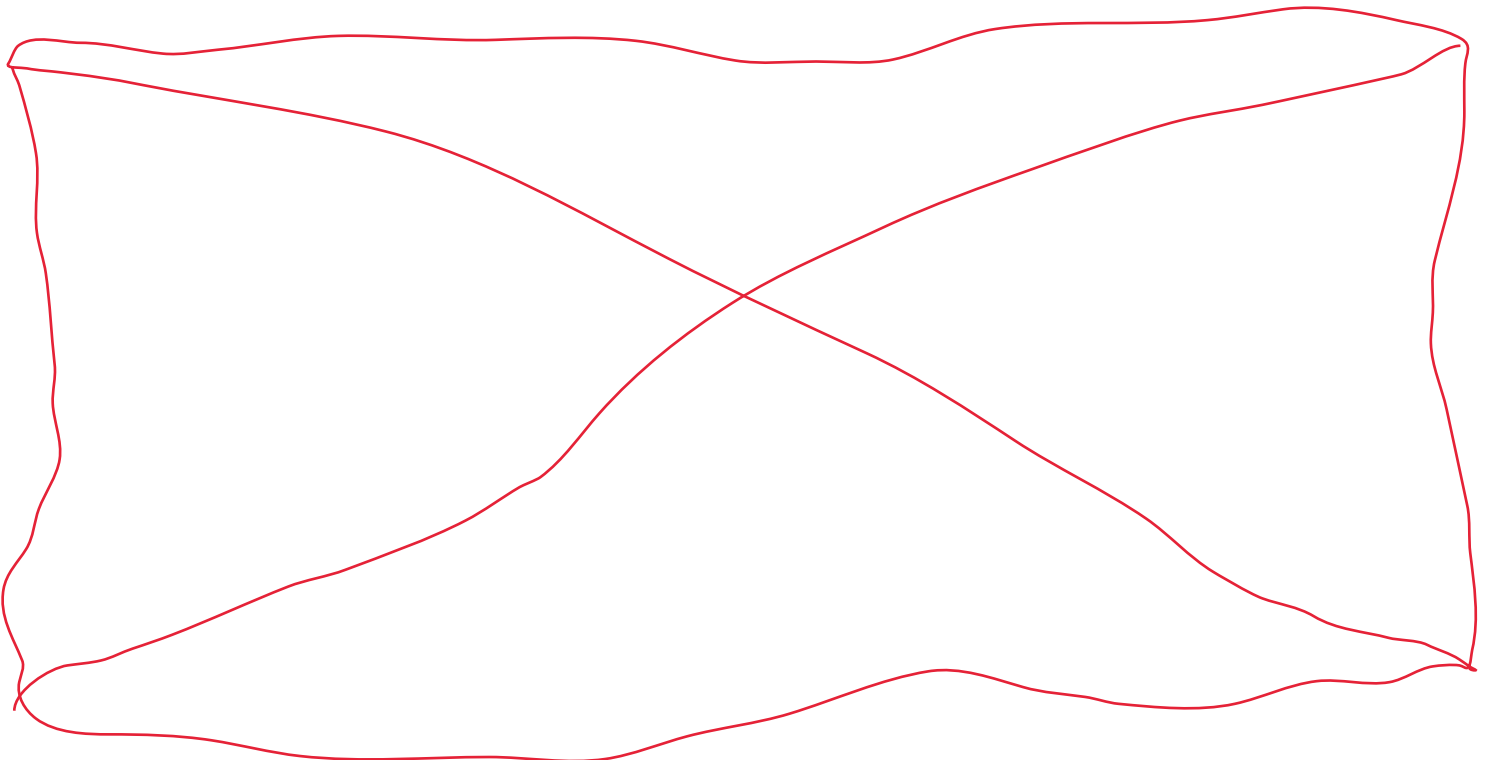
Figure 24: OC48 Original and sampled data sets Boxplot - 0.001s time frame

#### 4.1.2 Aggregate Variance

Looking at Table 6 containing the Aggregate Variance algorithm Hurst parameter estimations, the results show a very strong long range dependence on both the original and the sampled data sets except for the Systematic Time-based technique (SystT) at coarser granularity time frames (0.1s) in which the resulting parameter implies the sampled data series is closer to a random walk. The count-based sampling techniques are the best performers at the 0.1s and 0.01s time frames. At **THE** finer time frame of 0.001s all the data sets of different sampling techniques show estimations close to the original data set, with some of the best performing techniques being MuST, SystC(1/8) and the SystT techniques. Note that the correlation coefficient is very high, on all time frames and data sets.

Table 6: OC48 Aggregate Variance - Hurst parameter estimation and Correlation Coefficient

	0.1S	0.01S	0.001S
Original	0.857(86.88%)	0.923(96.38%)	0.908(93.15%)
LP(100/200)	0.649(99.01%)	0.671(94.96%)	0.771(94.23%)
MuST(200/500)	0.635(96.38%)	0.833(95.55%)	0.836(92.73%)
RandC(1/100)	0.814(94.38%)	0.837(94.50%)	0.740(93.92%)
SystC(1/8)	0.851(88.41%)	0.907(96.01%)	0.876(91.89%)
SystC(1/100)	0.815(94.50%)	0.835(94.58%)	0.727(94.96%)
SystC(1/256)	0.769(95.60%)	0.798(94.89%)	0.741(94.78%)
SystT(100/500)	0.429(96.86%)	0.611(97.51%)	0.776(91.35%)
SystT(200/500)	0.431(96.29%)	0.691(94.70%)	0.830(96.10%)
SystT(500/1500)	0.499(98.81%)	0.721(95.00%)	0.874(93.90%)
SystT(500/3500)	0.512(98.41%)	0.701(95.69%)	0.865(94.66%)



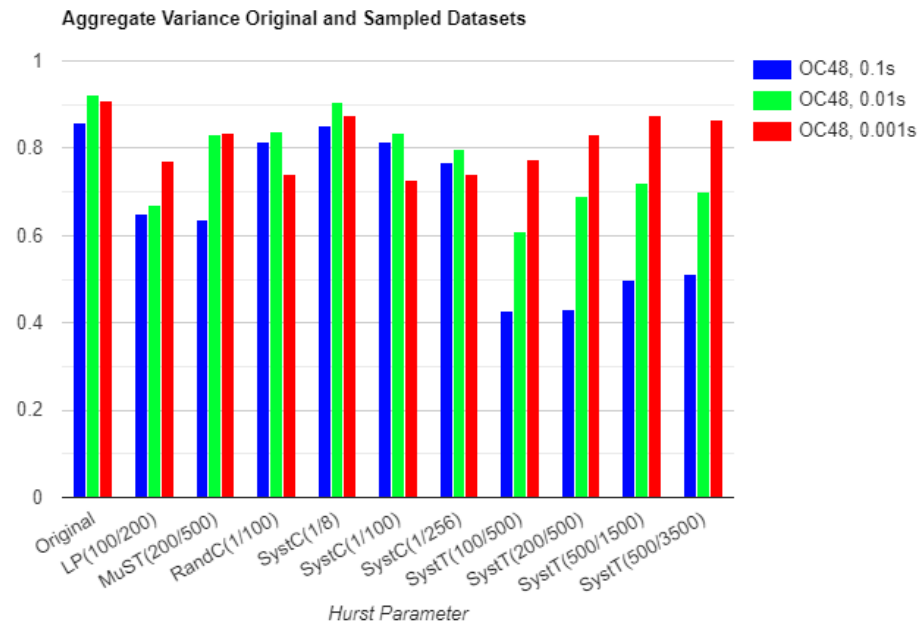


Figure 25: Bar chart of OC48 Aggregate Variance Hurst estimation per time frames

### 4.1.3 R/S

Looking at the results presented in Table 7 the R/S algorithm indicates LRD on all of the original data sets, showing stronger long range dependence at the coarser granularity of the 0.1s time frame. The sampling techniques show varying results with for example LP performing well at the 0.1s time frame and 0.001s. The SystC(1/8) data set shows very close results to the original one at all time frames and therefore being the best performing technique and interval, The time-based techniques start performing better at the 0.001s time frame.



Table 7: OC48 R/S - Hurst parameter estimation

	0.1s	0.01s	0.001s
Original	0.793	0.658	0.621
LP(100/200)	0.747	0.409	0.585
MuST(200/500)	0.281	0.614	0.626
RandC(1/100)	0.613	0.564	0.542
SystC(1/8)	0.741	0.614	0.575
SystC(1/100)	0.617	0.565	0.542
SystC(1/256)	0.590	0.555	0.550
SystT(100/500)	0.473	0.412	0.514
SystT(200/500)	0.456	0.368	0.558
SystT(500/1500)	0.426	0.426	0.598
SystT(500/3500)	0.440	0.460	0.608

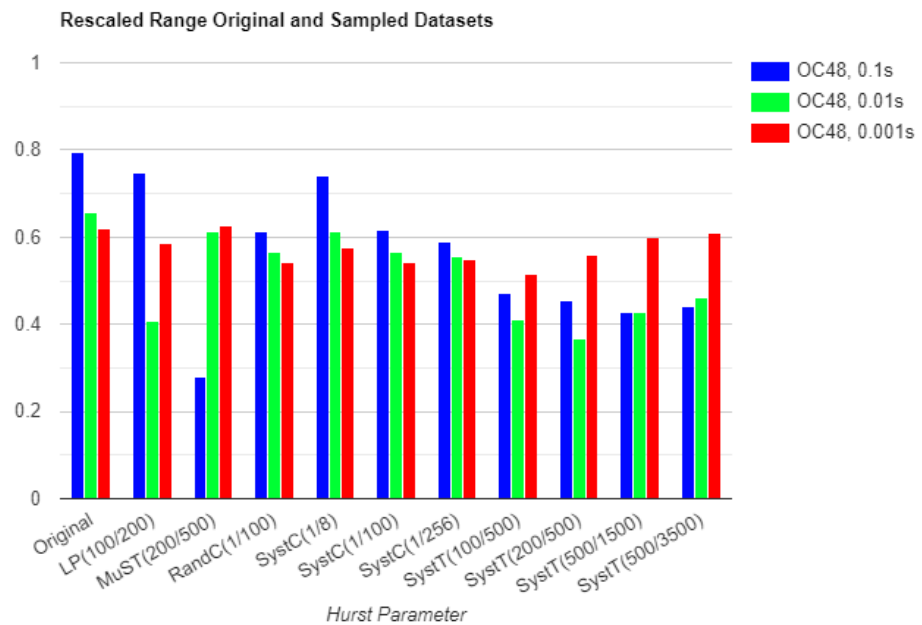


Figure 26: Bar chart of OC48 Rescaled Range Hurst estimation per time frames

#### 4.1.4 Periodogram

According to the author of [19] the Periodogram algorithms can present some values over 1 because in some cases non-stationarities are so complex, that conventional models fail to accommodate them, which can result in estimates of the Hurst parameter greater than or equal to 1, as shown in the following Table 8 with the original data set at the 0.1s time frame.

Once again as the R/S algorithm previously described, the Periodogram algorithm shows stronger LRD at the 0.1s time frame for all the data sets tested, except for the SystT technique. At the finer granularities the series starts to follow a random-walk as the Hurst parameter estimation is closer to 0.5, no longer presenting strong LRD. Still, the sampling techniques that show the closest results relative to the original are of the MuST and SystC(1/8) data sets.

Table 8: OC48 Periodogram - Hurst parameter estimation

	0.1s	0.01s	0.001s
Original	1.053	0.663	0.589
LP(100/200)	0.775	0.485	0.565
MuST(200/500)	0.728	0.669	0.582
RandC(1/100)	0.822	0.539	0.507
SystC(1/8)	0.950	0.618	0.542
SystC(1/100)	0.731	0.543	0.505
SystC(1/256)	0.666	0.524	0.517
SystT(100/500)	0.540	0.371	0.514
SystT(200/500)	0.518	0.440	0.543
SystT(500/1500)	0.462	0.495	0.570
SystT(500/3500)	0.444	0.500	0.577

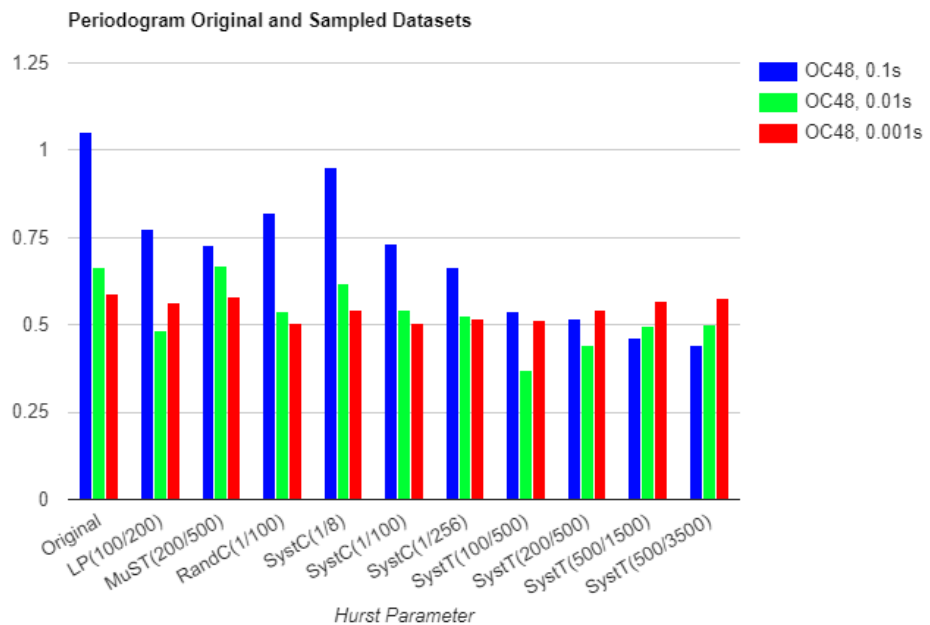


Figure 27: Bar chart of OC48 Periodogram Hurst estimation per time frames

## 4.2 OC-192

For the OC192 and with 0.1s time frame data set only  $N = 32768$  samples could be imported to the SELFIS tool, for the 0.01s  $N = 262144$  and finally for the 0.001s time frame  $N = 2097152$ . Note that the OC192 data set should be 3721s long which is roughly one hour of network traffic data.

### 4.2.1 Basic Statistics

The following results are very similar to the ones previously obtained in 4.1.1 when comparing the different time frames, only differing on the higher throughput obtained due to the OC192 network carrier setup, which makes use of more recent technologies than OC48, and therefore capable of higher data transfer rate.

#### 0.1s time frame data set

The following Table 9 show the basic statistics for the OC192 data set for the 0.1s time frame. Best performing sampling techniques are the count-based ones and MuST in terms of measuring throughput relative to the original data set.

Table 9: OC192 Basic statistics - 0.1s interval per sample

	Mean	Throughput	Variance	Std.Deviation	Skewness
Original	17.85	4524Mbps	0.015	0.126	-17.75
LP(100/200)	16.77	1535Mbps	1.391	1.179	-2.131
MuST(200/500)	17.70	3891Mbps	0.408	0.639	-5.319
RandC(1/100)	13.24	45(4500)Mbps	0.014	0.119	-8.316
SystC(1/8)	15.65	501(4008)Mbps	0.015	0.125	-16.17
SystC(1/100)	13.23	45(4500)Mbps	0.014	0.120	-8.692
SystC(1/256)	12.30	18(4608)Mbps	0.015	0.125	-7.187
SystT(100/500)	13.67	69Mbps	27.94	5.286	-0.713
SystT(200/500)	15.14	301Mbps	21.80	4.669	-1.398
SystT(500/1500)	16.17	842Mbps	16.21	4.027	-2.106
SystT(500/3500)	16.12	801Mbps	17.09	4.134	-2.028

Looking at Fig. 28, the massive variance of the SystT sampling techniques is noticeable mainly with the SystT(100/500) and SystT(200/500). Count-based techniques on the other hand present very low variance, comparable to the original data set, however they present much lower throughput results, as only a small percentage of the packets (depending on the interval set) is actually captured.

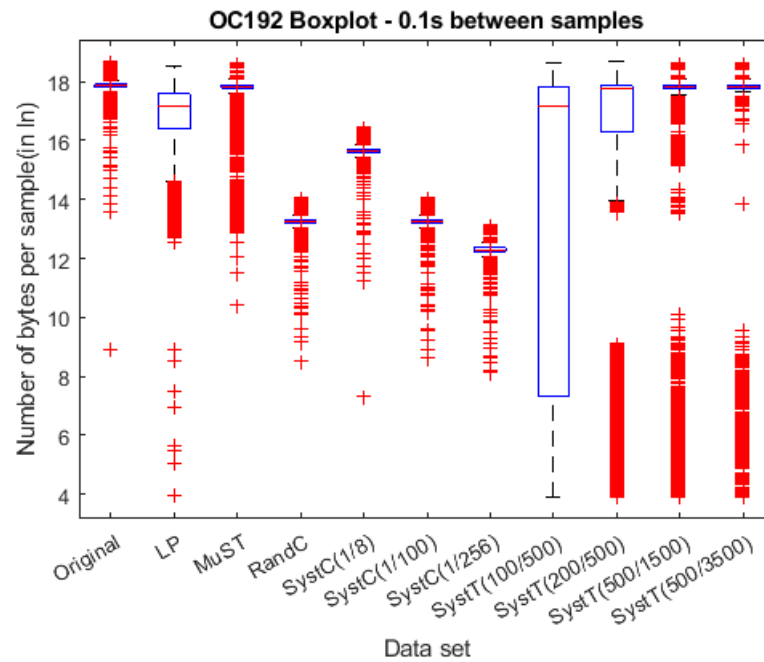


Figure 28: OC192 Original and sampled data sets Boxplot - 0.1s time frame

#### 0.01s time frame data set

The following Table 10 show the basic statistics for the OC192 data set for the 0.01s time frame. Best performing sampling techniques are again all of the count-based techniques and MuST for measuring throughput, be it instantaneous or mean throughput. The time-based techniques including SystT start performing better relative to the 0.1s time frame, with MuST presenting 0.048 variance to the original data set with 0.024 of variance.

Table 10: OC192 Basic statistics - 0.01s interval per sample

	Mean	Throughput	Variance	Std.Deviation	Skewness
Original	15.53	4446Mbps	0.024	0.155	-0.240
LP(100/200)	15.36	3748Mbps	0.380	0.616	-6.204
MuST(200/500)	15.53	4446Mbps	0.048	0.220	-9.547
RandC(1/100)	10.93	45(4500)Mbps	0.032	0.181	0.153
SystC(1/8)	13.34	497(3976)Mbps	0.024	0.157	-0.068
SystC(1/100)	10.92	44(4400)Mbps	0.033	0.182	-0.022
SystC(1/256)	9.979	17(4352)Mbps	0.048	0.221	-0.238
SystT(100/500)	14.86	2272Mbps	5.576	2.361	-3.437
SystT(200/500)	15.21	3226Mbps	1.652	1.652	-5.311
SystT(500/1500)	15.37	3786Mbps	1.503	1.226	-7.417
SystT(500/3500)	15.37	3786Mbps	1.505	1.227	-7.427

Looking at Fig. 29, it is noticeable how much closer in performance all types of sampling techniques are of each other. SystT still presents relatively high variance, mainly with the SystT(100/500) sampling interval.

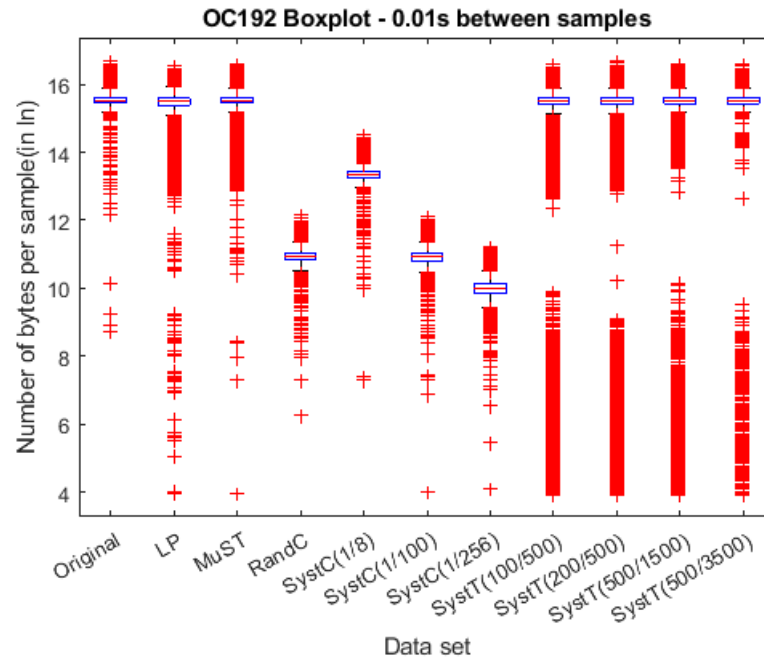


Figure 29: OC192 Original and sampled data sets Boxplot - 0.01s time frame

#### *0.001s time frame data set*

The following Table 11 show the basic statistics for the 0.001s time frame OC192 data set which follow the same trends as the OC48 statistics at the same time frame. The time-based techniques perform the best with LP and MuST estimating throughput very closely to the number obtained with the original data set, with very low variance as well. Interestingly the count-based techniques performed worse relatively to the other two time frames tested, but still maintaining good estimations.

Looking at the boxplot of Fig. 30, and as it happens with the OC48 data set on the time frame of 0.001s, the time-based techniques perform better when compared to the 0.1s and 0.01s time frames. MuST presents a good throughput estimation and 0.059 variance to the original of 0.054. Main difference to the OC48 is the fact that count-based techniques (SystC(1/8)) still show results close to the original data set at this time frame for the OC192 data set.

Table 11: OC192 Basic statistics - 0.001s interval per sample

	Mean	Throughput	Variance	Std.Deviation	Skewness
Original	13.21	4366Mbps	0.054	0.232	0.553
LP(100/200)	13.19	4280Mbps	0.175	0.419	-9.907
MuST(200/500)	13.22	4410Mbps	0.059	0.244	-1.503
RandC(1/100)	8.549	41(4100)Mbps	0.212	0.460	-1.324
SystC(1/8)	11.01	484(3872)Mbps	0.062	0.249	0.330
SystC(1/100)	8.541	41(4100)Mbps	0.206	0.454	-1.313
SystC(1/256)	7.426	13(3328)Mbps	0.829	0.910	-1.636
SystT(100/500)	13.16	4153Mbps	0.500	0.707	-9.061
SystT(200/500)	13.19	4279Mbps	0.271	0.520	-11.02
SystT(500/1500)	13.20	4322Mbps	0.144	0.379	-11.95
SystT(500/3500)	13.20	4322Mbps	0.141	0.376	-11.99

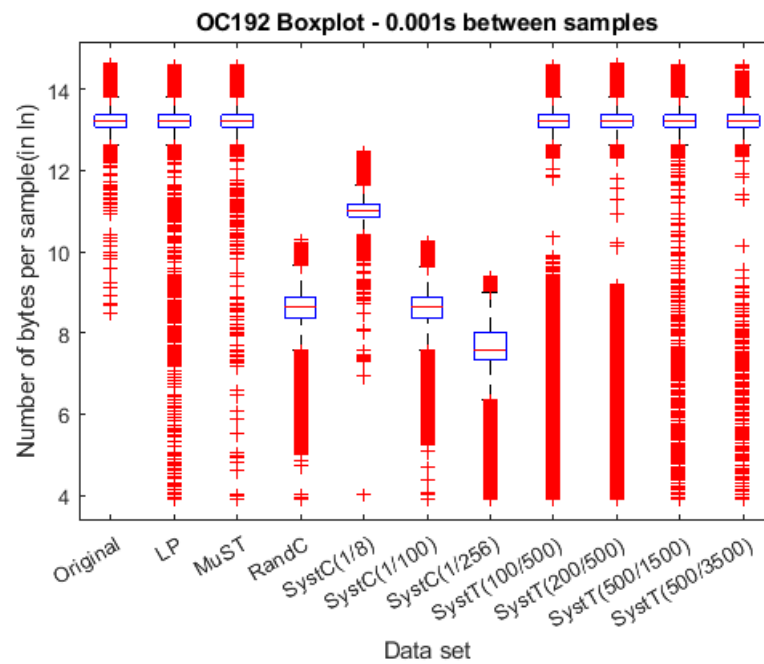


Figure 30: OC192 Original and sampled data sets Boxplot - 0.001s time frame

#### 4.2.2 Aggregate Variance

As seen before with the OC48 data sets and with the Aggregate Variance algorithm, the series presents LRD, at all three different time frames. Best performing sampling techniques that come closest to the original data set are MuST, and the count-based ones.

Table 12: OC192 Aggregate Variance - Hurst parameter estimation and Correlation Coefficient

	0.1s	0.01s	0.001s
Original	0.784(99.17%)	0.828(98.40%)	0.835(98.93%)
LP(100/200)	0.560(98.89%)	0.584(99.55%)	0.660(96.60%)
MuST(200/500)	0.729(90.33%)	0.770(99.58%)	0.801(99.24%)
RandC(1/100)	0.793(99.12%)	0.819(98.71%)	0.804(98.69%)
SystC(1/8)	0.785(99.15%)	0.827(98.42%)	0.832(98.97%)
SystC(1/100)	0.832(98.97%)	0.818(98.75%)	0.804(98.75%)
SystC(1/256)	0.770(98.27%)	0.794(98.16%)	0.787(99.23%)
SystT(100/500)	0.822(90.38%)	0.720(95.32%)	0.714(98.58%)
SystT(200/500)	0.664(96.34%)	0.523(88.33%)	0.752(98.89%)
SystT(500/1500)	0.671(96.08%)	0.626(98.01%)	0.775(99.46%)
SystT(500/3500)	0.382(99.56%)	0.558(98.80%)	0.755(99.64%)

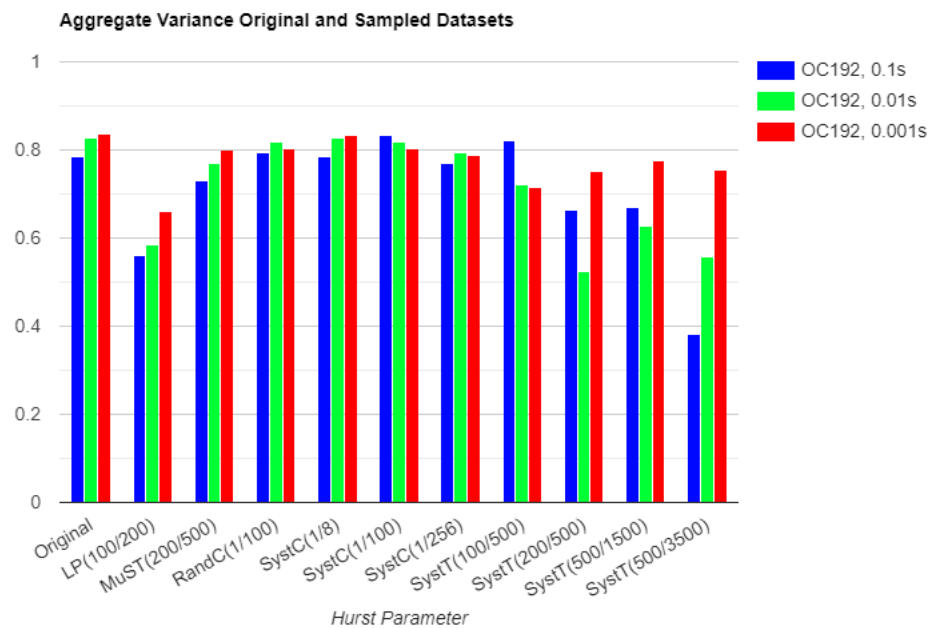


Figure 31: Bar chart of OC192 Aggregate Variance Hurst estimation per time frames

#### 4.2.3 R/S

Hurst estimations using the R/S algorithm show LRD looking at the original data set results from Table 13, however the sampling technique data sets that still maintain that dependence are the count-based ones (RandC and SystC), mainly at coarser granularities.

Table 13: OC192 R/S - Hurst parameter estimation

	0.1s	0.01s	0.001s
Original	0.800	0.740	0.732
LP(100/200)	0.598	0.474	0.658
MuST(200/500)	0.445	0.650	0.774
RandC(1/100)	0.793	0.703	0.650
SystC(1/8)	0.798	0.734	0.724
SystC(1/100)	0.790	0.703	0.650
SystC(1/256)	0.763	0.670	0.602
SystT(100/500)	0.586	0.297	0.458
SystT(200/500)	0.432	0.315	0.545
SystT(500/1500)	0.340	0.452	0.689
SystT(500/3500)	0.366	0.492	0.718

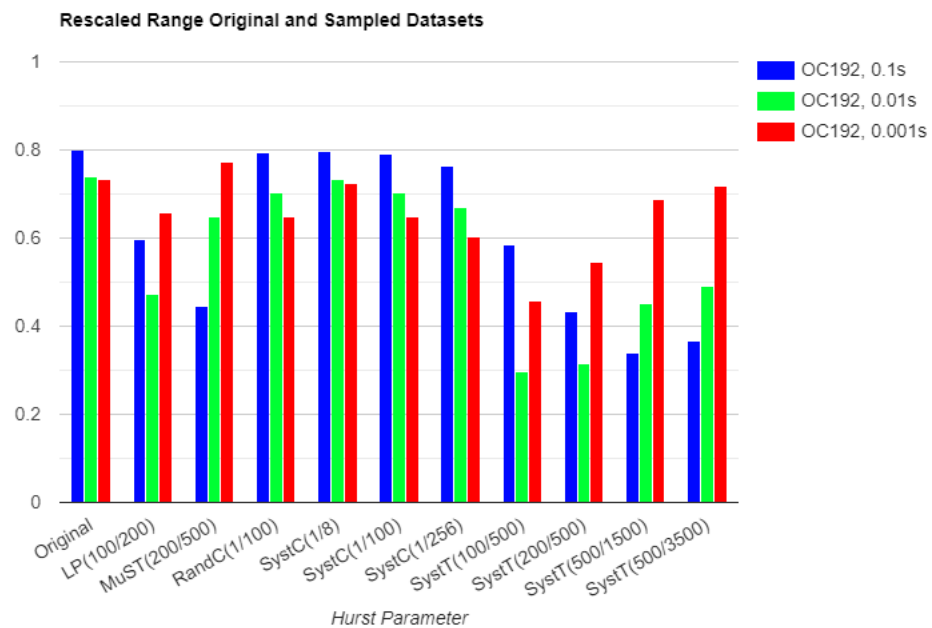


Figure 32: Bar chart of OC192 Rescaled Range Hurst estimation per time frames

#### 4.2.4 Periodogram

Using the Periodogram algorithm on OC192 data set with different time frames, show stronger LRD at 0.1s as it did with the OC48 data set. Best performing sampling techniques are the count-based ones, mainly RandC and SystC(1/8). Time-based techniques start performing closer to the Original data set at the 0.001s time frame.



Table 14: OC192 Periodogram - Hurst parameter estimation

	0.1s	0.01s	0.001s
Original	0.857	0.676	0.755
LP(100/200)	0.650	0.553	0.716
MuST(200/500)	0.575	0.662	0.762
RandC(1/100)	0.867	0.661	0.649
SystC(1/8)	0.859	0.676	0.742
SystC(1/100)	0.742	0.658	0.648
SystC(1/256)	0.951	0.778	0.838
SystT(100/500)	1.035	0.435	0.652
SystT(200/500)	0.974	0.497	0.687
SystT(500/1500)	0.618	0.516	0.713
SystT(500/3500)	0.536	0.508	0.726

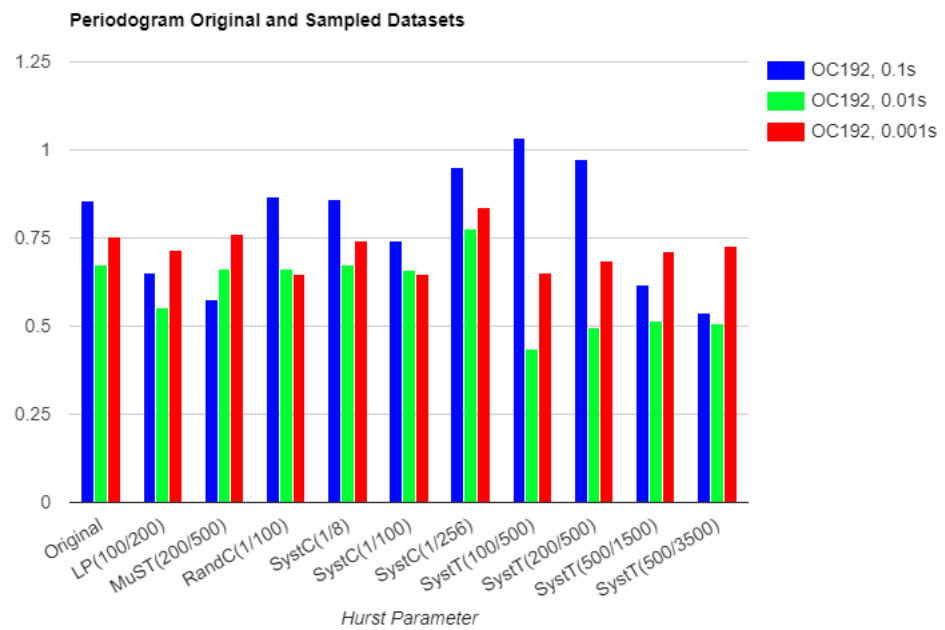


Figure 33: Bar chart of OC192 Periodogram Hurst estimation per time frames

#### 4.2.5 Autocorrelation Function

The Autocorrelation function (ACF) previously described in 2.2.4. According to the developer of the SELFIS software analysis tool [33], shows the value of the autocorrelation coefficient for different time lags  $k$  as previously described in 15.

The following Figures 34, 35 and 36 show the graphical representation of the ACF of typical long-range dependent processes as they slowly decay to zero for the Original, Random count-based and Systematic count-based techniques. The ACF for the 0.01s and 0.001s time frames present the same behaviour even though they are not visually represented here.

*Original data set - 0.1s time frame*

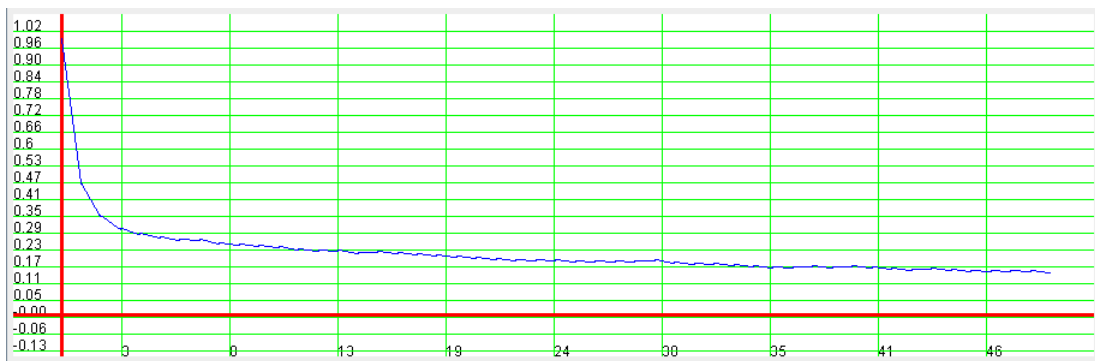


Figure 34: ACF graph of OC192 Original data set - 0.1s time frame

*Random count-based data set - 0.1s time frame*

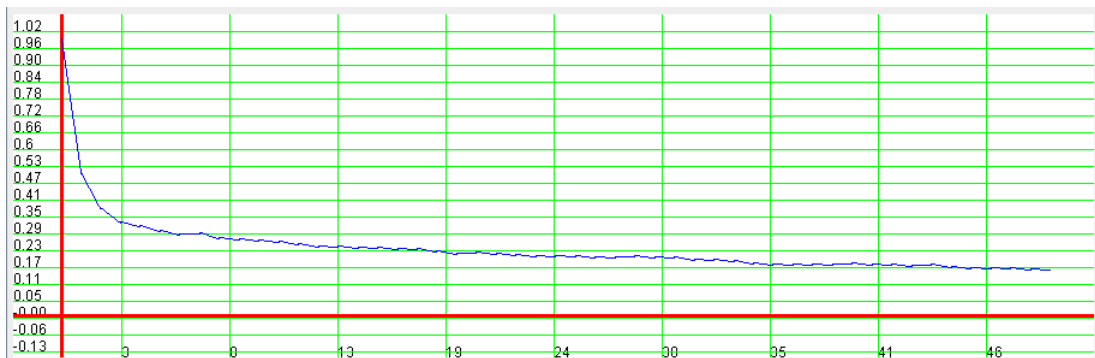


Figure 35: ACF graph of OC192 RandC data set - 0.1s time frame

*Systematic count-based data set - 0.1s time frame*

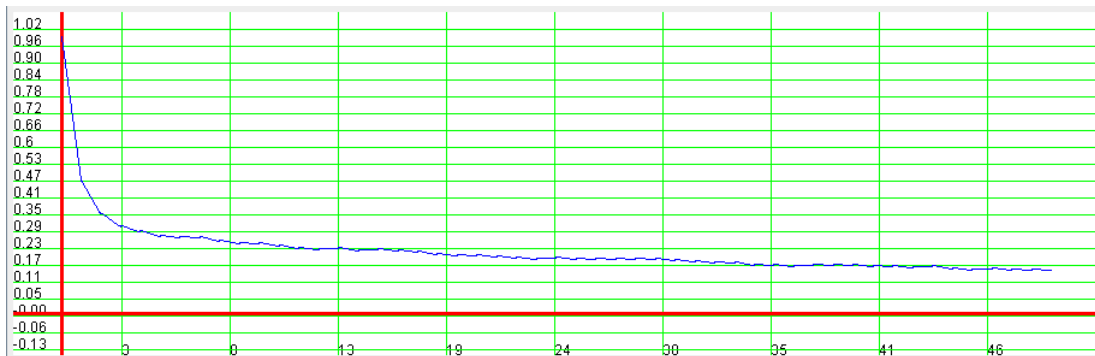


Figure 36: ACF graph of OC192 SystC data set - 0.1s time frame

However for the time-based techniques represented by 37, 38 and 39, the behaviour is a lot different to both the count-based techniques and each other, possibly because of the periodicity introduced by the time-based techniques.

*LP data set - 0.1s time frame*

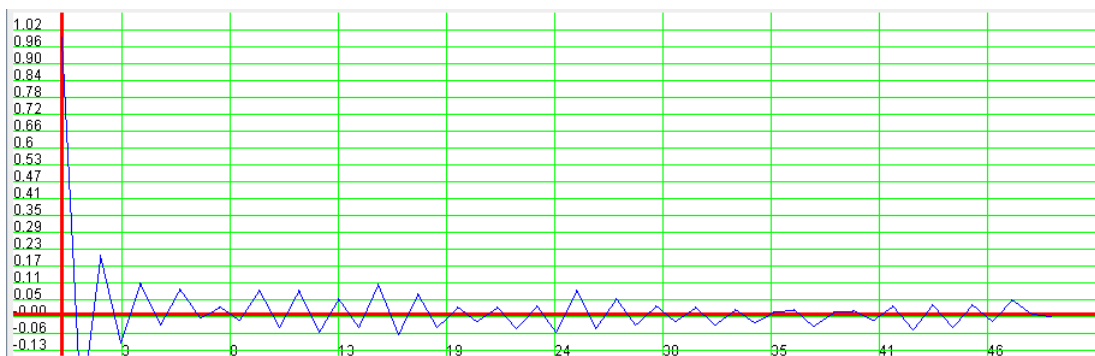


Figure 37: ACF graph of OC192 LP data set - 0.1s time frame

*MuST data set - 0.1s time frame*

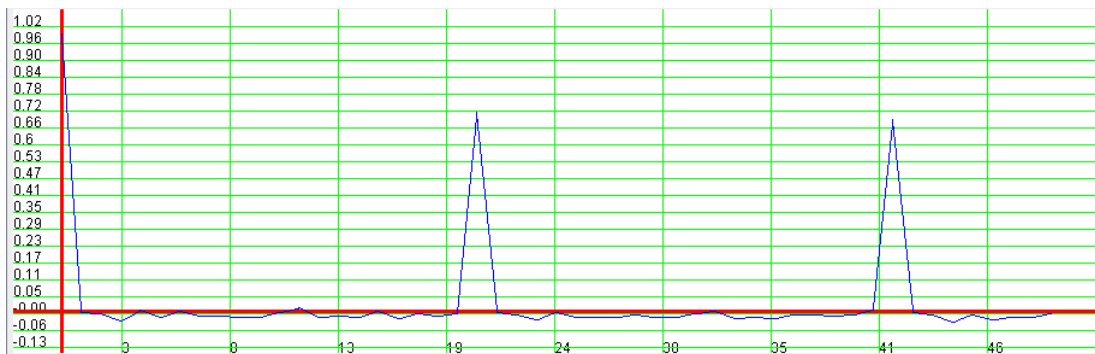


Figure 38: ACF graph of OC192 MuST data set - 0.1s time frame

*Systematic time based data set - 0.1s time frame*

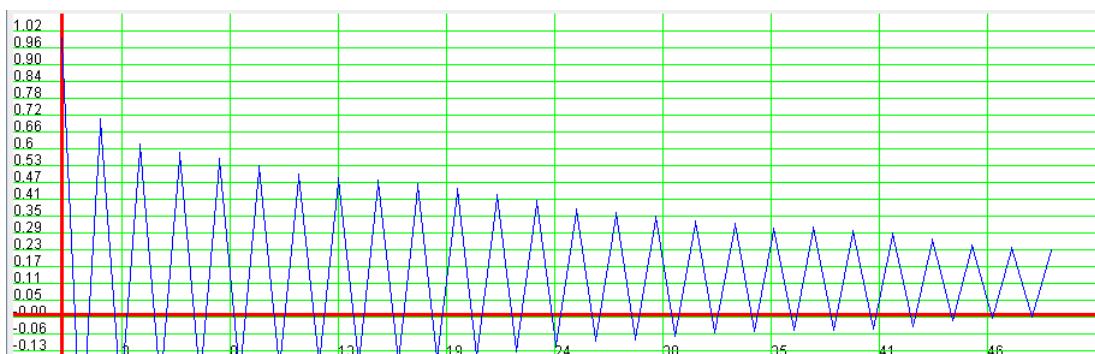


Figure 39: ACF graph of OC192 SystT data set - 0.1s time frame

### 4.3 SUMMARY

According to the author of [15] the presence of periodicity in a time series may have an influence on the estimation of the long memory (long-range dependence) parameter  $H$ , as some estimators falsely detect the presence of long-range dependence when periodicity is present.

Both the OC48 and OC192 present similar results with all the different sampling techniques tested, both in regards of basic statistics and Hurst estimation parameter.

#### 4.3.1 Data Volume

Note that even though the OC<sub>192</sub> capture was one hour long, it still used a lot more disk space volume in comparison to the OC<sub>48</sub> capture which was three hours long, as the data transfer rates and bandwidth of OC<sub>192</sub> are a lot higher.

The next Fig. 40 demonstrates how much space each data set occupies, including the sampled ones.

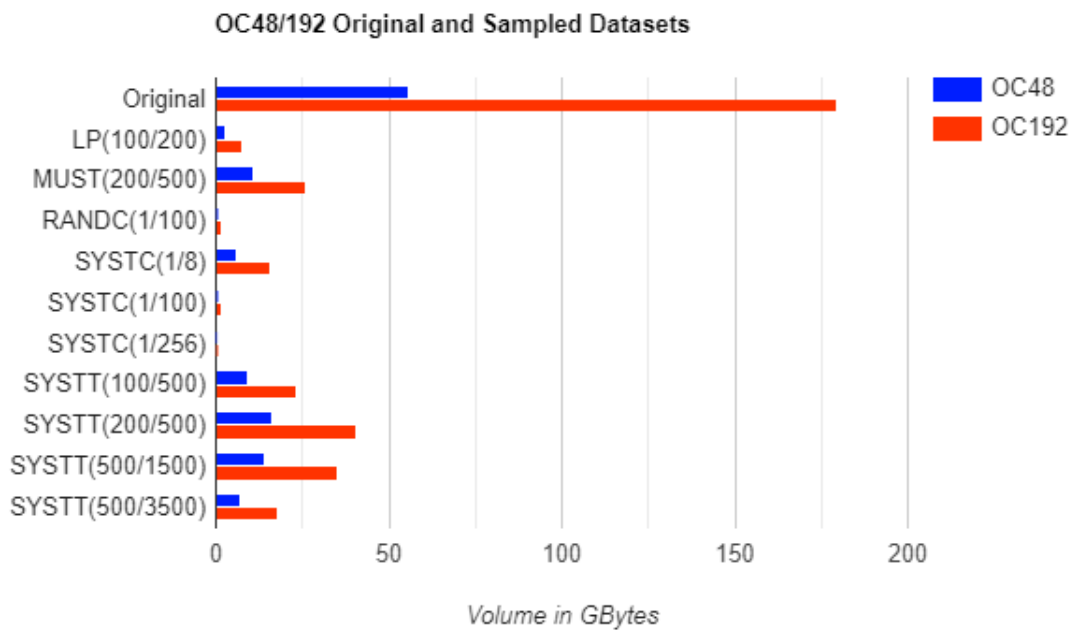


Figure 40: Original and sampled OC data sets volume

#### 4.3.2 Throughput and Hurst Estimations

In terms of estimating throughput and the Hurst parameter, with all three algorithms and three different time frames tested from both OC<sub>48</sub> and OC<sub>192</sub>, the best performing techniques are very dependent on the time frame of the data set. As expected for the Hurst estimations, the original series showed LRD or strong memory. However, only some sampled data sets still maintained that property of Internet traffic.

For the 0.1s time frame, the best performing techniques were the count-based ones (RandC and SystC), which like the original data set, present very low variance and deviation, necessary for instantaneous throughput estimations to be accurate.

However, as the time granularity becomes smaller (0.01s and 0.001s time frames), the count-based techniques start performing worse, inversely to what happens with the time-based techniques. These start to shine in particular with the MuST which was the most consistent technique, presenting good results, even at the 0.1s time frame. The sampling technique of SystC with the sampling interval of  $(1/8)$  was also a very consistent data set in both throughput and Hurst estimations, even at the finer granularity time frames, which is impressive considering it is one of the sampling techniques that use the least amount of data volume, compared to the time-based techniques.

---

## CONCLUSION

---

The work developed in this dissertation was mainly focused on the statistical properties of sampled Internet traffic when compared to the original data and in which way those properties are affected due to the loss of data after applying the sampling process with the advantage of lower computational resources being needed.

Depending on which type of sampling processes used and whether they are count-based or time-based as well as the time frame of the data set is used, the results obtained with the estimation algorithms are different. The resulting sampled data sets also use a different amount of disk space, with time-based techniques generally using more when compared to the count-based techniques.

Analysing the tests and results obtained described in Chapter 4, it is still possible to gather accurate information for network traffic analysis after applying the sampling process to the original traffic data, both in terms of estimating throughput, as well as in determining if the sampled series still maintain the property of long range dependence (LRD).

SystC(1/8) and MuST had some of the consistent results both in throughput and Hurst parameter estimations, presenting very low variance in the number of bytes per sample of the time frame selected. Therefore these data sets provide good estimations on instantaneous and mean throughput and on the Hurst parameter. Time-based techniques (LP, MuST and SystT) also start performing better at the finer granularity time frames, inversely to what happens to count-based techniques, mainly at less frequent sampling intervals as, for example, SystC(1/100) or SystC(1/256).

### 5.1 FUTURE WORK

Additional work associated can still be developed to better understand how the sampling process affects the results of the Hurst parameter and other statistics. For example, it is possible to apply a smoothing function on the original and sampled data sets or a low-pass filter which is a filter that passes signals with a frequency lower than a selected cutoff frequency and attenuates signals with frequencies higher than the cutoff frequency to

eliminate certain periodicities of the original and sampled signals, as well as using different data sets and algorithm combinations.

It is also possible to obtain some other statistics and the usage of different intervals for each of the sampling techniques than the ones tested in this dissertation to evaluate how much that parameter affects the results.



---

## BIBLIOGRAPHY

---

- [1] João Marco C. Silva, Paulo Carvalho, and Solange Rito Lima. Computational weight of network traffic sampling techniques. In *2014 IEEE Symposium on Computers and Communications (ISCC)*, pages 1–6, 2014. doi: 10.1109/ISCC.2014.6912467.
- [2] S. Bregni. Compared accuracy evaluation of estimators of traffic long-range dependence. In *2014 IEEE Latin-America Conference on Communications (LATINCOM)*, pages 1–5, 2014. doi: 10.1109/LATINCOM.2014.7041868.
- [3] T. Karagiannis, M. Molle, and M. Faloutsos. Long-range dependence ten years of internet traffic modeling. *IEEE Internet Computing*, 8(5):57–64, 2004. doi: 10.1109/MIC.2004.46.
- [4] H. E. HURST. Long-term storage capacity of reservoirs. *Trans. Amer. Soc. Civil Eng.*, 116: 770–799, 1951. URL <https://ci.nii.ac.jp/naid/10024052023/en/>.
- [5] AJ Lawrance and NT Kottegoda. Stochastic modelling of riverflow time series. *Journal of the Royal Statistical Society: Series A (General)*, 140(1):1–31, 1977.
- [6] John Haslett and Adrian E. Raftery. Space-time modelling with long-memory dependence: Assessing ireland’s wind power resource. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 38(1):1–50, 1989. ISSN 00359254, 14679876. URL <http://www.jstor.org/stable/2347679>.
- [7] A Montanari, R Rosso, and MS Taqqu. A seasonal fractional differenced arima model: An application to the river Nile monthly flows at aswan. *preprint*, 1995.
- [8] Alberto Montanari, Renzo Rosso, and Murad S Taqqu. Some long-run properties of rainfall records in Italy. *Journal of Geophysical Research: Atmospheres*, 101(D23):29431–29438, 1996.
- [9] Alberto Montanari, Renzo Rosso, and Murad S Taqqu. Fractionally differenced arima models applied to hydrologic time series: Identification, estimation, and simulation. *Water resources research*, 33(5):1035–1044, 1997.
- [10] Clive William John Granger and Paul Newbold. *Forecasting economic time series*. Academic Press, 2014.
- [11] Richard T Baillie. Long memory processes and fractional integration in econometrics. *Journal of econometrics*, 73(1):5–59, 1996.

- [12] Zhuanxin Ding and Clive WJ Granger. Modeling volatility persistence of speculative returns: a new approach. *Journal of econometrics*, 73(1):185–215, 1996.
- [13] Jan Beran, Robert Sherman, Murad S Taqqu, and Walter Willinger. Long-range dependence in variable-bit-rate video traffic. *IEEE Transactions on communications*, 43(2/3/4):1566–1579, 1995.
- [14] Walter Willinger, Murad S Taqqu, Will E Leland, and Daniel V Wilson. Self-similarity in high-speed packet traffic: analysis and modeling of ethernet traffic measurements. *Statistical science*, pages 67–85, 1995.
- [15] A. Montanari, M.S. Taqqu, and V. Teverovsky. Estimating long-range dependence in the presence of periodicity: An empirical study. *Mathematical and Computer Modelling*, 29(10): 217–228, 1999. ISSN 0895-7177. doi: [https://doi.org/10.1016/S0895-7177\(99\)00104-1](https://doi.org/10.1016/S0895-7177(99)00104-1). URL <https://www.sciencedirect.com/science/article/pii/S0895717799001041>.
- [16] T. Zseby, Mayra Molina, Nick Duffield, Saverio Niccolini, and Frederic Raspall. Sampling and filtering techniques for ip packet selection. 01 2008.
- [17] J. M. C. Silva, P. Carvalho, and S. R. Lima. Analysing traffic flows through sampling: A comparative study. In *2015 IEEE Symposium on Computers and Communication (ISCC)*, pages 341–346, 2015. doi: 10.1109/ISCC.2015.7405538.
- [18] Baek-Young Choi and Supratik Bhattacharyya. Observations on cisco sampled netflow. *SIGMETRICS Performance Evaluation Review*, 33:18–23, 12 2005. doi: 10.1145/1111572.1111579.
- [19] Cheolwoo Park, Felix Hernández-Campos, J.S. Marron, and F. Smith. Long-range dependence in a changing internet traffic mix. *Computer Networks*, 48:401–422, 06 2005. doi: 10.1016/j.comnet.2004.11.018.
- [20] V. Paxson and S. Floyd. Wide area traffic: the failure of poisson modeling. *IEEE/ACM Transactions on Networking*, 3(3):226–244, 1995. doi: 10.1109/90.392383.
- [21] MURAD S. TAQQU, VADIM TEVEROVSKY, and WALTER WILLINGER. Estimators for long-range dependence: An empirical study. *Fractals*, 03(04):785–798, 1995. doi: 10.1142/S0218348X95000692. URL <https://doi.org/10.1142/S0218348X95000692>.
- [22] M. Krunz. On the limitations of the variance-time test for inference of long-range dependence. In *Proceedings IEEE INFOCOM 2001. Conference on Computer Communications. Twentieth Annual Joint Conference of the IEEE Computer and Communications Society (Cat. No.01CH37213)*, volume 3, pages 1254–1260 vol.3, 2001. doi: 10.1109/INFOCOM.2001.916620.

- [23] H. E. Hurst, R. P. Black, and Y. M. Simaika. *Long-term storage : an experimental study / by H.E. Hurst, R.P. Black, Y.M. Simaika*. Constable London, 1965.
- [24] W. J. Riley and C. A. Greenhall. Power law noise identification using the lag 1 autocorrelation. In *2004 18th European Frequency and Time Forum (EFTF 2004)*, pages 576–580, 2004. doi: 10.1049/cp:20040932.
- [25] Kihong Park and Walter Willinger. *Self-Similar Network Traffic: An Overview*, pages 1–38. 01 2002. ISBN 0471319740. doi: 10.1002/047120644X.ch1.
- [26] G. W. Wornell and A. V. Oppenheim. Estimation of fractal signals from noisy measurements using wavelets. *IEEE Transactions on Signal Processing*, 40(3):611–623, 1992. doi: 10.1109/78.120804.
- [27] P. Abry and D. Veitch. Wavelet analysis of long-range-dependent traffic. *IEEE Transactions on Information Theory*, 44(1):2–15, 1998. doi: 10.1109/18.650984.
- [28] D. Veitch and P. Abry. A wavelet-based joint estimator of the parameters of long-range dependence. *IEEE Transactions on Information Theory*, 45(3):878–897, 1999. doi: 10.1109/18.761330.
- [29] Stefano Bregni. *Characterization and Modelling of Clocks*, pages 203 – 281. 04 2002. ISBN 9780470845882. doi: 10.1002/0470845880.ch5.
- [30] Jan Beran. *Statistics for Long-Memory Processes*. Routledge, 1994.
- [31] Esther Stroe-Kunold, Tatjana Stadnitski, Joachim Werner, and Simone Braun. Estimating long-range dependence in time series: An evaluation of estimators implemented in r. *Behavior research methods*, 41:909–23, 09 2009. doi: 10.3758/BRM.41.3.909.
- [32] Gerald Combs. Wireshark <https://www.wireshark.org/>.
- [33] Thomas Karagiannis. Selfis: A short tutorial. 12 2002.
- [34] MATLAB. The MathWorks Inc. <https://www.mathworks.com/products/matlab.html>, Natick, Massachusetts, 2021.