Universidade do Minho
Escola de Engenharia

João Marco Cardoso da Silva

A Modular Traffic Sampling Architecture for Flexible Network Measurements

Outubro de 2015

A Modular Traffic Sampling Architecture for Flexible Network Measurements
João Marco Cardoso da Silva
UMinho | 2015

Universidade do Minho
Escola de Engenharia

João Marco Cardoso da Silva

A Modular Traffic Sampling Architecture for Flexible Network Measurements

Dissertação de Mestrado
Programa Doutoral em Informática

Trabalho Efetuado sob a orientação da
Professora Maria Solange Pires Ferreira Rito Lima

Outubro de 2015

DECLARAÇÃO DE INTEGRIDADE

Declaro ter atuado com integridade na elaboração da presente tese.

Confirmo que em todo o trabalho conducente à sua elaboração não recorri à prática de plágio ou a qualquer forma de falsificação de resultados.

Mais declaro que tomei conhecimento integral do Código de Conduta Ética da Universidade do Minho.

Universidade do Minho, 30 de outubro de 2015

Nome completo: João Marco Cardoso da Silva

Assinatura: ______________________________

Acknowledgments

To my family, especially my parents Antônio Marcos and Kátia Valéria, and my sister Jamille Silva for the unconditional support, love and understanding during these years I have been away from home.

To Professor Solange Rito Lima for supervising this work and for her continuos advice, guidance and support along the last years, since my master at University of Minho.

To Professor Paulo Carvalho for his continuous encouragement and assistance in many aspects of this work.

To Alice Balbé for her everyday understanding, support, encouragement and love.

To my good friends for many cheerful and enjoyable moments during this work-intensive period, specially Bernardo Reis, Rafael Pereira, Andriza Saraiva, Hermes Pimentel, Humberto Longo, Jorge Guedes and Ricardo Martini.

Thank you for supporting me in achieving this goal.

Abstract

The massive traffic volumes and the heterogeneity of services in today's networks urge for flexible, yet simple measurement solutions to assist network management tasks, without impairing network performance. To turn treatable tasks requiring traffic analysis, sampling the traffic has become mandatory, triggering substantial research in the area. In fact, multiple sampling techniques have been proposed to assist network engineering tasks, each one targeting specific measurement goals and traffic scenarios. Despite that, there is still a lack of an encompassing solution able to support the flexible deployment of these techniques in production networks.

In this context, this research work proposes a modular traffic sampling architecture able to foster the flexible design and deployment of efficient measurement strategies. The architecture is composed of three layers *i.e.*, *management plane*, *control plane* and *data plane* covering key components to achieve versatile and lightweight measurements in diverse traffic scenarios and measurement activities. The flexibility and modularity in deploying different sampling strategies relies upon a novel taxonomy of sampling techniques, in which, current and emerging techniques are identified regarding their inner characteristics - *granularity*, *selection trigger* and *selection scheme*.

Following the proposed taxonomy, a sampling framework prototype has been developed and used as an experimental implementation of the proposed architecture, providing a fair environment to assess and compare sampling techniques under distinct measurement scenarios. Supported by the sampling framework, distinct techniques have been evaluated regarding their performance in balancing the computational burden and the accuracy in supporting traffic workload estimation and flow analysis. The results have demonstrated the relevance and applicability of the proposed architecture, revealing that a modular and configurable approach to sampling is a step forward for improving sampling scope and efficiency.

Resumo

Os grandes volumes de tráfego e a heterogeneidade de serviços nas redes atuais requerem soluções de medição que sejam flexíveis e simples de modo a sustentar as tarefas de gestão de redes sem afetar o desempenho das mesmas. Para tornar tratável as tarefas que exigem análise de tráfego, tornou-se obrigatório recorrer a amostragem do tráfego, motivando uma investigação substancial na área. Como consequência, várias técnicas de amostragem foram propostas para auxiliar as tarefas de engenharia de redes, cada uma orientada a satisfazer objetivos de medição e cenários de tráfego específicos. Apesar disso, ainda não existe uma solução abrangente capaz de suportar a implantação flexível destas técnicas em redes de produção.

Neste contexto, este trabalho propõe uma arquitetura modular de amostragem de tráfego capaz de fomentar a concepção flexível e a implementação de estratégias eficientes de medição de tráfego. A arquitetura é composta por três camadas, nomeadamente, *camada de gestão*, *camada de controle* e *camada de dados*, cobrindo os principais componentes para alcançar versatilidade e baixo custo computacional em variados cenários de tráfego e atividades de medição. A flexibilidade e modularidade na implementação de diferentes técnicas de amostragem baseia-se numa nova taxonomia, na qual técnicas atuais e emergentes são identificadas de acordo com suas características internas - *granularidade*, *trigger de seleção* e *esquema de seleção.*

Seguindo a taxonomia proposta, um protótipo estruturando e agregando as diferentes técnicas de amostragem foi desenvolvido e utilizado na implementação experimental da arquitetura, permitindo avaliar e comparar as técnicas de amostragem em diversos cenários de medição. Suportado pelo protótipo desenvolvido, distintas técnicas foram avaliadas quanto ao seu desempenho em equilibrar a carga computacional e a acurácia na estimação do volume de tráfego e na análise de fluxos. Os resultados demonstraram a relevância e aplicabilidade da arquitetura de amostragem proposta, revelando que uma abordagem modular e configurável constitui um avanço no sentido de melhorar a eficiência na amostragem de tráfego.

Contents

List of Figures

List of Tables

List of Acronyms

API	Application Programming Interface
CAIDA	Center for Applied Internet Data Analysis
CIM	Common Information Model
CPU	Central Processing Unit
DDoS	Distributed Denial of Service
DMA	Direct Memory Access
DPI	Deep Packet Inspection
HTTPS	Hyper Text Transfer Protocol Secure
IANA	Internet Assigned Numbers Authority
IDS	Intrusion Detection System
IETF	Internet Engineering Task Force
INE	Instituto Nacional de Estatística
IP	Internet Protocol
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6
IPFIX	Internet Protocol Flow Information Export
ISP	Internet Service Provider
JVM	Java Virtual Machine
KDE	Kernel Density Estimation
LP	Linear Prediction
LP_F	Linear Prediction_Flow-level
MIB	Management Information Base
MP	Measurement Point
MRE	Mean Relative Error
MSE	Mean Square Error
MTP	Mean Throughput
MTU	Maximum Transmission Unit
MuST	Multiadaptive Sampling Technique
MuST_F	Multiadaptive Sampling Technique_Flow-level

NIC	Network Interface Card
OC	Optical Carrier
SDN	Software Defined Networks
PIB	Policy Information Base
PSAMP	Packet SAMPling
QoS	Quality of Service
RandC	Uniform-Random Count-based
RandC_F	Uniform-Random Count-based_Flow-level
RFC	Request for Comments
RME	Relative Mean Error
SCTP	Stream Control Transmission Protocol
SILK	System for Internet-Level Knowledge
SLA	Service Level Agreement
SMI	Structure of Management Information
SNMP	Simple Network Management Protocol
SystC	Systematic Count-based
SystC_F	Systematic Count-based_Flow-level
SystEvt	Systematic Event-based
SystEvt_F	Systematic Event-based_Flow-level
SystT	Systematic Time-based
SystT_F	Systematic Time-based_Flow-level
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
UTC	Coordinated Universal Time
Uminho	University of Minho
Xcas	Explicit Multi-Unicast
XML	Extensible Markup Language

Chapter 1

Introduction

The growth in size and heterogeneity of today's communication networks has brought huge challenges to network planning and management activities. The need for efficient monitoring solutions, being crucial to assist service providers and network managers in these activities, is further stressed when considering aspects such as service convergence, mobility, virtualization and ubiquity, which are expected to coexist in a seamless network environment.

One of the most used and versatile strategies designed to assist network monitoring is traffic measurement. Traffic measurement techniques can be applied in real, emulated or simulated networks and the Measurement Points (MPs) can be deployed directly in the network nodes, in dedicated equipment or in general-purpose devices connected to the network under analysis.

The massive traffic volume traversing high-capacity links, the distinct accuracy requirements associated with service types or monitoring objectives to fulfill, sometimes in a near real-time basis and requiring minimal interference in the network operation, are the main challenges when designing a measurement methodology.

Traffic measurements can adopt *active* or *passive* methodologies. The active methodology resorts to the use of intrusive traffic by injecting probe packets into the network for measurement purposes. Although generally more versatile and easy to deploy, this method imposes caution with the traffic overhead in order to avoid disturbing the network operation. Moreover, different activities might require distinct probe packet features, which increase the risk of overloading the network with extra traffic.

Passive measurement methodologies adopt a non-intrusive approach, considering only the real traffic of the network under analysis. These techniques typically use specific devices or mechanisms embedded in network nodes. The major difficulty associated with the usage of passive measurements is the volume of traffic involved, resulting in high resources requirements for processing, storage and transmission of data [1].

Aiming to cope with the tension between reducing the volume of data involved in passive measurements and supplying sufficiently detailed information for the various monitoring activities, three strategies are commonly employed - *aggregation*, *filtering* and *sampling* [2].

Aggregation involves grouping related traffic by values in the packet header, such as the same IP (Internet Protocol) source and destination addresses [3], or by temporal distribution, such as all packets traversing the MP at a specific time interval. A common usage of aggregation for providing compact data summaries is flow accounting, which consists in the incremental collection of statistics about a group of packets that share common properties [4]. Although effective in reducing the volume of data, this strategy leads to a loss of visibility regarding all traffic.

Filtering uses a deterministic selection of packets based on the packet content, on the treatment of the packet at the monitored node or on a function involving both of them [1], discarding all packets in which the selection key does not match. In this way, filtering is particularly useful for monitoring a subset of traffic of interest, once that subset has been identified [2].

Sampling consists of selecting a subset of packets that will allow to estimate parameters about all traffic, with compatible degrees of accuracy, avoiding processing it completely [5]. Sampling differs from aggregation and filtering as it is not required to know the traffic features of interest in advance, thereby allowing its wide usage on multiple measurement tasks in presence of diverse traffic types. Therefore, packet sampling has become mandatory for effective passive network measurements, especially in the network core, reducing the amount of data to a manageable size [6].

Whilst the usage of sampling reduces the volume of traffic for analysis and improves measurements versatility when compared with aggregation and filtering, the wide adoption of sampling-based measurements still faces challenging aspects. In particular, (*i*) choosing the best sampling technique toward the traffic characteristics or monitoring goals; (*ii*) the accuracy in estimating the underlying metrics; and (*iii*) the impact of processing and storage involved in traffic sampling processes as well as the transport of the measured data, are open issues deserving further study.

1.1 Motivation and objectives

Despite passive packet sampling being an appropriate solution both to decrease the massive volume of data processed, stored and transmitted when compared with measuring all network traffic, and to reduce the interference in the network operation caused by active measurements, this strategy retains some characteristics and constraints that drive substantial research and a large number of new proposals.

As network traffic is heavily dynamic, estimating parameters about the network accurately and frequently near real time becomes a non trivial task [7]. The sampling technique chosen must be able to collect samples that represent the overall traffic behavior. However, a sample used to estimate a particular parameter correctly may not be ideal for a different parameter or traffic type [2].

Packet sampling techniques are usually driven by schemes that rule packet capturing at MPs: *systematic* techniques rely on a deterministic function based on the packet position in time or in space into the incoming traffic; *random* techniques resort to a probabilistic function in order to decide which packets will be selected to compose the sample; and *adaptive* techniques introduce some intelligence aiming at selecting the most relevant part of the traffic traversing the measurement point [6].

Based on the principles presented above, many recent works have proposed sampling techniques and policies able to achieve better results regarding the accuracy in metric estimation or the reduction of computational overhead for various measurement tasks. However, most of these techniques usually address a unique network monitoring activity, such as SLA (Service Level Agreement) compliance [7][8][9], QoS (Quality of Service) parameters [9][10][11], anomaly detection [12][13] and others, as detailed in Chapter 2.

Despite the undeniable importance of traffic sampling to assist large scale passive measurements and the performance differences of each approach when applied to distinct traffic types or measurement goals, most of the main tools able to perform packet sampling (*e.g.*, Cisco Netflow and sFlow) only support a small number of sampling techniques, without covering relevant classic and new proposals. These limitations are mainly related to the complexity in achieving an acceptable trade-off between providing several sampling techniques and the computational requirements involved in each technique. In this way, this work aims to address a crucial question toward effective passive measurements in today's and future networks:

How packet sampling measurement systems should be designed in order to attend different monitoring objectives considering diverse traffic characteristics and network scenarios?

The answer to this question also involves addressing fundamental aspects for the ample adoption of packet sampling, namely: (i) *compatibility* with currently deployed measurement systems; (ii) *flexibility* to accommodate new measurement goals and traffic characteristics; and (iii) *lightweight operation* in order to minimize interference with the normal network tasks. Hence,

the main objective of this work is to devise an encompassing, flexible and lightweight traffic sampling architecture aiming at fostering the design and deployment of efficient sampling strategies for diverse traffic scenarios and measurement goals.

The efficiency of sampling strategies is mainly related to the ability to balance accuracy in metric estimations and computational resources required.

To pursue this objective, this work is focused on understanding sampling techniques through its constituent parts, rather than a closed unit, aiming to identify its specific characteristics, advantages and constraints. This will allow to guide the design of sampling systems able to optimize their performance by exploiting the most suitable features for a specific measurement purpose or traffic type.

A more detailed view of the problem to solve leads to the following objectives:

- survey existing techniques and tools for network and communication services measurement based on packet sampling, identifying their main characteristics, advantages and limitations;
- identify and classify the main sampling techniques regarding their inner characteristics, providing thus a modular view of their components, and the performance achieved in estimating metrics related to the measurement tasks they support;
- conceive and specify an encompassing sampling-based measurement architecture able to drive the design of efficient measurement systems supported by a flexible and modular structure;
- implement a prototype of the proposed architecture as a configurable framework able to deploy multiple sampling techniques based on their previous modular classification;

- provide a proof-of-concept environment aiming at evaluating the effectiveness of the proposed architecture and the trade-off regarding accuracy in metric estimations, volume of data involved and computational requirements for various sampling techniques in diverse network scenarios.

1.2 Research methodology

The research methodology to pursue the main objective of this work comprises three well defined steps somehow evinced by the list of objectives defined above. Firstly, considering the main aspects related to traffic measurements supported by packet sampling, relevant literature covering the major developments and methods is surveyed. This bibliographic search and review allows a comprehensive analysis of the current state-of-the-art in this area, grounding the proposal of a packet sampling taxonomy able to describe current and emerging techniques in a modular and hierarchical structure. Secondly, a new sampling architecture is devised and its main subjacent entities are specified in order to accomplish the goals related to compatibility, flexibility and lightweight operation. Aiming at providing a proof-of-concept of the new proposal, a prototype of the sampling architecture is implemented as a modular framework and evaluated toward its applicability in supporting traffic measurements in diverse scenarios efficiently. Performing tests in real traffic scenarios allows to assess the suitability of distinct sampling strategies in manifold measurement tasks taking into account aspects such as accuracy, volume of data involved and computational burden.

1.3 Summary of main contributions

Although a full discussion on this thesis outcomes is included in the concluding chapter, this section provides a brief overview of its main contributions. Taking into account the initial objectives defined above, these contributions are summarized as follows:

- proposal of a new multiadaptive sampling technique able to improve network measurements efficiency over distinct traffic conditions by self-adjusting the packet sampling policy according to the ongoing network activity [14] [15] [16] [17];

- proposal of a taxonomy of sampling techniques providing a modular view of the current approaches which can be explored both to adjust sampling configuration to specific measurement requirements and to enhance the performance of network measurement systems [18] [19];

- proposal of an encompassing and flexible packet sampling architecture addressing key components to foster the deployment of versatile and lightweight measurement strategies in diverse traffic scenarios and measurement activities [18] [20];
- development of a sampling framework following a multilayer design in order to sustain the modular deployment of current and forthcoming sampling techniques [21] [22];
- fair comparative assessment of various sampling techniques and policies regarding their computational weight (*i.e.*, CPU load, memory consumption and volume of data involved) [21], the accuracy in estimating traffic workload [20] and to support flow analysis [23] [22].

1.4 Dissertation layout

This dissertation is structured in eight chapters reflecting the research work carried out facing the objectives outlined in Section 1.1.

In the present Chapter 1 - *Introduction* - a first positioning of the reader in the area of research is provided, highlighting current trends and evolution. Then, the motivation for this thesis is justified and the main objectives of the work are defined. The main contributions to the research field are also summarized. The dissertation layout is included here in order to provide a global view of the full document, regarding its contents and organization.

In Chapter 2 - *Packet Sampling for Network Measurements* - is firstly introduced a consistent terminology of the main concepts related to packet sampling. Then, an encompassing overview of the techniques currently used for packet sampling is presented, detailing both standard specifications, widely deployed in measurement tools, and new proposals, commonly designed to improve specific measurement goals. The chapter also summarizes related work in which the performance and impact of packet sampling are addressed.

In Chapter 3 - *Multiadaptive Sampling Technique* - is proposed a new self-adaptive sampling technique based on linear prediction, which aims at reducing the computational weight in the network measurements without compromising estimation accuracy. For this purpose, the packet selection process considers the levels of network activity, being configured to reduce the measurement impact when the network activity increases and the measurement process tends to be heavier. The multiadaptive behavior of this proposal is achieved considering both the sampling interval and the sample size as adaptive parameters, bounded by thresholds that guarantee the representativeness of samples in capturing network behavior.

In Chapter 4 - *Taxonomy of Packet Sampling Techniques* - a novel taxonomy is proposed defining 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. Describing these components in a modular and hierarchical structure allows to foster the flexible and simple deployment of a comprehensive number of techniques in order to sustain the design of efficient measurement strategies. The chapter also demonstrates the comprehensiveness of the proposal by describing several existing techniques through the taxonomy.

In Chapter 5 - *Sampling-Based Measurement Architecture* - is presented and specified a complete architecture of measurement systems based on the modular deployment of sampling techniques. The architecture is composed by three layers, *management plane*, *control plane* and *data plane*, covering all the components involved in traffic measurements. Aspects such as mapping the measurement needs into the most suitable sampling technique and operational parameters are discussed, including the decision on the packet fields of interest, aggregation level and exporting format according to the network scenario and measurement goals to fulfill.

In Chapter 6 - *Sampling Framework Prototype* - is detailed the technological aspects and the development of a functional sampling framework used as a proof-of-concept of the flexibility introduced by the sampling taxonomy and architecture. This framework provides a fair environment to perform comparative assessments involving different sampling techniques in diverse traffic scenarios. The chapter also presents an experimental distributed sampling-based measurement system running in a large-scale network and a complete example of the proposed architecture operation.

In Chapter 7 - *Test Scenarios and Results* - several test scenarios are introduced in order to (*i*) demonstrate the versatility of the proposed architecture in providing a flexible solution able to accommodate diverse sampling strategies and therefore, fostering the development of efficient measurement strategies; (*ii*) evaluate the performance of different sampling techniques when applied to activities usually supported by sampled measurements, *i.e.*, *traffic workload* and *flow analysis*. The performance is assessed comparing the accuracy in estimating related metrics and the computational requirements (*i.e.*, CPU load, memory consumption and volume of data involved) in performing such techniques.

In Chapter 8 - *Conclusions* - an overview of the present research work is presented, concluding on to what extent the objectives defined initially have been accomplished. This chapter also presents the main contributions of this thesis, providing future research directions based on a critical analysis of the work carried out.

Chapter 2

Packet Sampling for Network Measurements

The usage of packet sampling aiming at fostering network measurements is not a recent research subject. The initial efforts addressing sampling techniques for statistical analysis of computer networks were mainly focused on QoS of communication systems, traffic accounting and characterization [24] [25] [26]. These early research works also have produced methods to categorize the sampling techniques, starting from Amer and Cassel [27] and further evolving to a framework standardized as a Request for Comments (RFC) [1] by the Packet SAMPling (PSAMP) Working Group of the Internet Engineering Task Force (IETF) [6].

Simultaneously with the development of high-speed network infrastructures and the diversification of communication services, the usage of packet sampling has also increased significantly, leading to the support of manifold tasks related to network measurements. As illustrated in Figure 2.1, examples of these tasks include: *network management* involving short, medium and long term planning and management of network operation, maintenance and provisioning of network services [28][29][30][31]; *traffic engineering* involving performance optimization, traffic characterization, traffic modeling and control [32][33][25][6][34]; *performance evaluation* of protocols and management tools, network reliability and fault tolerance [35][36][9]; *network security*, including detection of anomalies, intrusion, botnet and Distributed Denial of Service (DDoS) attacks [37][38][39][40][41][42]; *SLA compliance*, where auditing tools may resort to network sampling for measuring and report service levels [43] [44]; *QoS control*, aiming at measuring parameters such as delay, jitter and packet loss [45][10][46][47]. Most of the above cited techniques are conceived resorting to modifications and/or composition of basic approaches that compose classical sampling techniques. These techniques are described in Section 2.2.

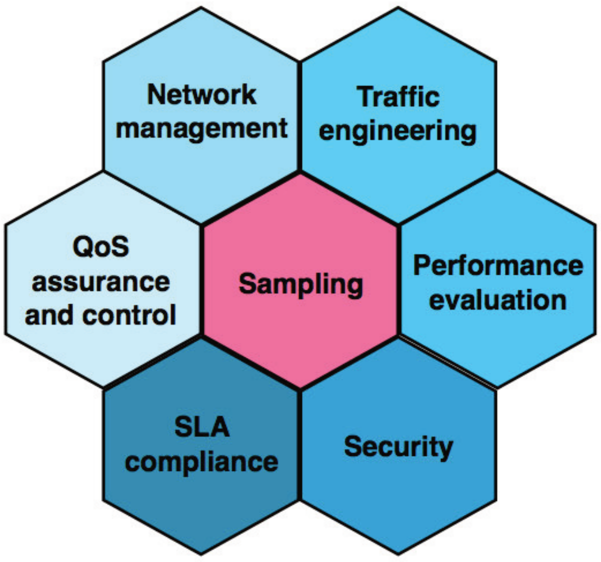


Figure 2.1: *Measurement tasks supported by packet sampling*

A large coverage of the related literature shows that some of the basic concepts related to packet sampling are frequently presented in ambiguous way. To avoid inconsistencies, this chapter firstly presents, in Section 2.1, the terminology assumed throughout this work. Section 2.2 presents the sampling technique categorization defined in [1]. These categories are further extended by the stratified techniques, in Section 2.3, and adaptive techniques, in Section 2.4. Then, in Section 2.5, is presented an overview of the practical application of packet sampling, the performance achieved in specific measurement tasks and the availability of these techniques in current sampling tools.

2.1 Sampling terminology

Aiming at providing a consistent terminology of the main concepts related to packet sampling, the most common terms are assumed throughout this work in accordance with the following definitions.

- *Sample* - subset of network packets that are selected at the MP and then considered in the estimation of network parameters. This is also often referred as *sample event*, which consists in an individual action of selection and capture of packets from the stream under analysis;
- *Sample size* - number of packets or time interval in which all incoming packets at the MP are selected and captured to compose a sample. The sample size is

controlled by triggers, that are responsible for starting and finishing each sample taking into account the packet position into the stream or its timestamp at the MP;

- *Interval between samples* - number of packets or time interval in which all incoming packets are ignored for measurement purposes. Likewise the sample size, the manipulation of interval between samples also resorts to triggers.

Figure 2.2 illustrates the above concepts.

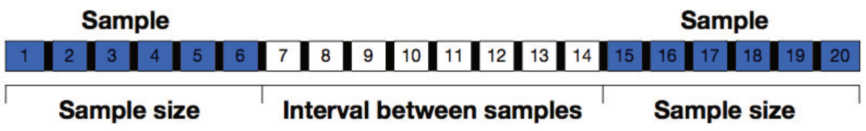


Figure 2.2: *Basic sampling concepts*

2.2 PSAMP - Packet SAMPling

The RFC 5475 [1] is a standard that describes techniques for sampling and filtering IP packets for measurement purposes. Although it does not recommend any particular implementation, the document provides a classification of these techniques according their process of selecting packets by dividing the algorithms into two classes: *content-independent* techniques and *content-dependent* techniques. The main difference between these classes is the necessity of accessing the packet content in order to take selection and capture decisions, which may impact on the computational resource requirements. Here, the packet content corresponds to the packet header (of any TCP/IP layers) and/or the packet payload.

2.2.1 Content-independent techniques

Inasmuch as these techniques do not inspect the packet content in order to rule the sampling decisions, they control the sample triggering process considering the position or the timestamp of the packet into the stream resorting to deterministic or (pseudo) random functions. Within the deterministic techniques are the *Systematic Count-based* and *Systematic Time-based* whereas the non-deterministic techniques encompass *Random n-out-of-N* and *Random Uniform Probabilistic*.

Systematic Count-based

In this technique, the process of ruling the starting point of a sample and its selection length (*sample size*) is driven by the spatial packet position (using packet counters) following a deterministic function that confers a periodic behavior. A common deployment of this technique (illustrated in Figure 2.3) is the periodic selection of every *k-th* packet of a stream. In this example, selecting every 5th packet, the sample size is equal to 1 and the interval between samples is 4. Due to its simplicity and low resource requirements, this technique has been widely deployed by network equipment vendors and industry [48].

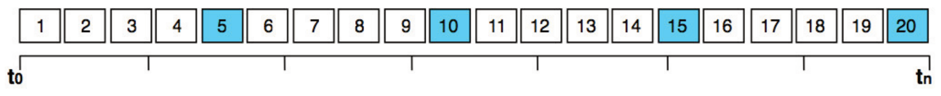


Figure 2.3: *Systematic Count-based*

Systematic Time-based

Likewise the systematic count-based, the systematic time-based technique also rules the sample size and interval between samples resorting to a deterministic function. However, in this case, the triggers are oriented to the arrival time of the packet at the MP. For instance, the selection and capture process takes every incoming packet during a specific time interval (*sample size*), ignoring every packet that reaches the MP during the interval between samples, as illustrated in Figure 2.4. In this example, the sample size is 100 milliseconds and the interval between samples is 200 milliseconds.

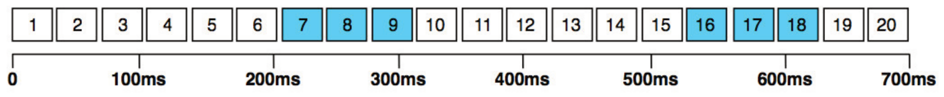


Figure 2.4: *Systematic Time-based*

Despite the relative simplicity of deployment, both systematic techniques face a risk of leading to biased measurement results, as the predictability of the periodic sampling enables deliberate manipulation or evasion [2].

Although RFC 5475 standard [1] only considers systematic techniques with samples equally spaced, it is also stated that even if their functions are not periodic (*e.g.*, the interval between samples varies over the time), they are still considered as a systematic technique. An example is the technique discussed in [2], in which the triggers that rule the packet capture follow a Poisson distribution based on the timestamp of the first packet observed.

Random n-out-of-N

The random techniques aim to avoid the risk of biased samples by ruling the packet selection and capture by a random process. The simplest and widely deployed mechanism consists in capturing n packets from a sequential stream of N packets (*n-out-of-N*). In this technique, a pseudorandom function generates n numbers between $[1, N]$, then the packets that have a position equal to one of the random numbers are selected and captured. Figure 2.5 illustrates an example where one packet is collected from every five incoming packets. It is important to notice that in this approach, all the N packets have the same probability p (with $p = n/N$) to be selected to compose the sample.

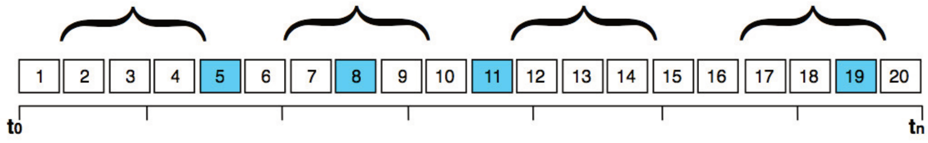


Figure 2.5: *Random n-out-of-N*

Random Uniform probabilistic

In the random uniform probabilistic technique, the decision for selecting packets to compose a sample is in accordance with a predefined uniform probabilistic function regardless the packet content. This means that all packets have the same probability to be selected. A deployment example consists in a count-driven technique in which the successive intervals between samples (with sample size equal to 1 packet) follow an independent random variable with distribution of mean $1/p$. Another uniform probabilistic implementation referred in [1] corresponds to a time-driven triggered technique where the interval between samples follows an exponential distribution.

2.2.2 Content-dependent techniques

This class of techniques involves the inspection of the packet content in order to drive sampling decisions. The techniques defined in [1] are *Random Non-uniform probabilistic*, *Random Non-uniform Flow-state* and *Hash-based sampling.*

Random Non-uniform probabilistic

In this technique, the packets do not have the same probability to be selected to compose a sample. For this, the function may consider the packet content or part of it in order, for instance, to increase the chance of selecting packets that are rare but are deemed important.

An example of this technique consists in selecting TCP packets containing the SYN flag active with a higher probability [6]. Figure 2.6 illustrates this technique, in which the packets with the SYN flag active are selected with probability $1/k$ and the remaining packets are selected with probability $1/p$. In this case, defining $p > k$ increases the probability of selecting packets from different flows.

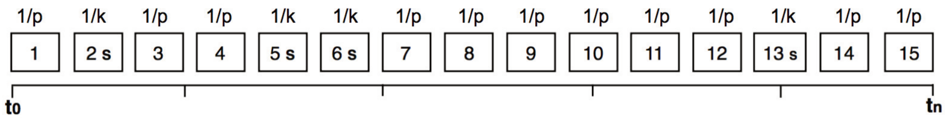


Figure 2.6: *Random Non-uniform*

Although RFC 5475 [1] classifies this technique as content-dependent, the same document also defines that the selection probability may be driven by the packet position, which does not require the packet content inspection.

Random Non-uniform Flow-state

In the sampling techniques previously presented, each packet from the stream under analysis is treated as an individual entity, with a particular chance to be selected for a sample according to the sampling scheme being used. In these approaches, for measurement tasks requiring traffic analysis based on flows, the packets are at first sampled indistinctly for subsequent classification, usually resorting to a flow key (*e.g.*, a 5-tuple composed by fields of the packet header, namely source and destination IP addresses, source and destination ports and type of protocol).

In this way, the *Random Non-uniform Flow-state* consists in ruling the packet selection by a specified state of the flow the packet belongs to or by the state of other flows being monitored. This involves classifying the incoming packet as belonging to a particular flow during the selection process. Therefore, despite of the captured packets are a sample of the total stream reaching the MP, this technique implies processing all packets in order to identify the flow key, which require additional computational resources. The standard also does not present any concrete method to define the flow state, leaving this definition to be adjusted toward specific goals.

Hash-based sampling

As mentioned before, the sampling and filtering techniques for IP packet selection are described in [1]. A *filtering* process is a deterministic selection of packets based on its content or treatment in a particular observation point. Although it usually involves a previous knowledge of which properties will the filter be driven for, the statistical properties of hash-based filtering allow it to be used as a random sampling emulator. The hash-based selection consists in applying a hash function to the packet content, or some portion of it, and then select all packets whose result falls into a defined range.

The decision on a hash function is widely addressed in [49] and must fulfill two conditions: *(i)* be invariant along the path; *(ii)* be variable among packets, in the sense that a small variation in the input packet (*e.g.*, IP address) will cause large changes in the hash output [2].

An example of sampling technique based on hash functions is presented in [29] (*i.e.*, *Trajectory sampling*). The technique uses the same hash function in different MPs in order to collect the same set of packets in the entire domain. This enables the reconstruction of the packet trajectories, thus assisting tasks that involve multipoint metric estimation of the same packets, such as end-to-end delay.

2.3 Stratified techniques

In addition to the characterization of sampling, the RFC 5475 [1] also introduces composite approaches, in which a sampling technique is defined by the application of different combination of sampling schemes in sequence. Thus, the sampling process is divided into multiple steps in order to increase the estimation accuracy of a specific parameter using the same number or less packets than those that would result from using a single scheme. One of these strategies is *stratified sampling.*

The basic idea behind stratified techniques is to group packets in strata according to an attribute and then sampling packets into each strata [2]. In this way, the stratified techniques require specifying classification rules for grouping the packets into subgroups (that can be performed in multiple steps) and the sampling scheme that will be used within the subgroups.

A deployment example is the *uniform stratified random sampling technique*, in which M consecutive incoming packets are grouped in a window, in which N packets (with $N < M$) are randomly sampled. Considering that the number of packets in each subgroup is constant, the selection probability for each packet is equal. Although this approach is similar to that presented in Section 2.2.1 (*i.e., random n-out-of-N*), here the packets may also be grouped by the arrival timestamps, as illustrated in Figure 2.7, or any other filtering process.

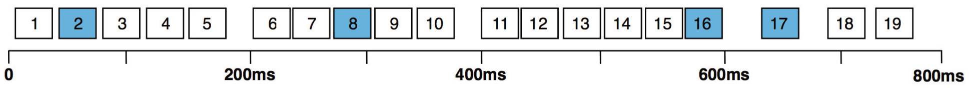


Figure 2.7: *Stratified sampling*

The stratification reduces the variance of single packet statistics if the variance between subgroups is greater than the variance within a subgroup. For instance, if the dividing characteristic is equal to the investigated characteristic, each element of the subgroup would be a perfect representative of that characteristic [1]. In this way, the stratification strategy (*i.e.*, the subgroup selection, dimensioning and sample size) are key aspects that impact on the performance of the technique. These aspects were investigated in [50], in which a *non-uniform stratified random sampling* is also introduced. This technique uses dynamic strategies for the adjustment of stratification boundaries applied to one-way delay measurements.

According to the outcomes in [50], although the stratified schemes can reduce significantly the costs for the sampling process (*i.e.*, the number of packets needed to achieve a particular level of confidence) when compared with single sampling schemes, the dynamic behavior only improves the accuracy if there are large changes of the trace characteristics between subgroups, which imply that the usage of more sophisticated methods does not necessarily lead to better results. This occurs due to the complexity of defining efficient methods to predict those characteristics for the subsequent subgroup.

2.4 Adaptive techniques

Beyond the schemes presented in [1], a large number of sampling techniques are able to adjust dynamically the selection of packets during the course of measurements. This involves introducing some intelligence into the sampling algorithm in order to biasing the capture towards packets carrying the most useful information for the specific measurement purpose. There is no a guideline or limit for the amount of intelligence embedded in the technique, nevertheless the solutions should keep the computational requirements under an acceptable limit [6]. The regular deployment of adaptive techniques usually resorts to controllers in order to rule the sampling policy toward a parameter observed in the network. These controllers may resort to predictive or non-predictive models and some examples of methods employed are *linear prediction*, *non-linear adjustment*, *fuzzy logic* or different *statistical* methods.

Regarding adaptive techniques, this doctoral work also introduces a new proposal (detailed in Chapter 3) driven by a predictive controller in order reduce the amount of data involved in the network measurements without compromising the estimation accuracy.

2.4.1 Linear prediction method

The linear prediction method proposed in [28], inspired by the congestion control protocol of TCP - Transmission Control Protocol, uses aspects observed in past samples to predict the future traffic behavior and then adjusting the sampling frequency to a suitable distribution in order to balance accuracy and computational overhead.

The main idea is to adjust the time interval between samples considering the level of activity observed in the network. When an increase in the network activity is detected, a shorter interval between samples is employed to measure the behavior of the network with higher accuracy, otherwise, when less activity is detected, the interval between samples is increased to reduce sampling overhead. The level of activity is assessed comparing the traffic volume predicted with the real value measured [28].

The method [28] considers the traffic volume observed in the last N samples to predict the traffic volume (x_p) in the next sample resorting to the equation:

$$x_p = x[N] + \frac{\Delta T_{current}}{N-1} \sum_{i=1}^{N-1} \left(\frac{x[i+1] - x[i]}{t[i+1] - t[i]} \right) \tag{2.1}$$

where, $x[i]$ is the traffic volume estimated in each previous sample i collected at the time $t[i]$, $x[N]$ is the traffic volume estimated in the most recent sample and $\Delta T_{current}$ is the time interval for the next sample schedule.

The predicted value x_p is then compared with the actual value estimated through the current *sample*. The comparison yields a rate of activity change m (presented in Equation 2.2) that is further applied to a set of rules to adjust the sampling interval to a new value ΔT_{next}, which is then used to schedule the next sampling event.

$$m = \left| \frac{x_p - x[N]}{sample - x[N]} \right| \tag{2.2}$$

The result produced by Equation 2.2 will be a value near the unity when the predicted behavior is close to the actual value. The value range is defined by $m_{min} < 1 < m_{max}$, with m_{min} and m_{max} bounded experimentally according to the traffic type and is interpreted as: *(i)* if m is below m_{max}, the volume of traffic is changing faster than predicted, so the sampling interval should be decreased; *(ii)* if m is above m_{max}, the traffic volume is changing more slowly than the prediction, so the sampling interval can be increased. The detailed rules to define the next sample interval are presented in Table 2.1 [28].

Table 2.1: *Rules to define the* ΔT_{next}

value of m	ΔT_{next}
$m < m_{min}$	$\Delta T_{next} = m\ \Delta T_{current}$
$m_{min} \leq m \leq m_{max}$	$\Delta T_{next} = \Delta T_{current}$
$m > m_{max}$	$\Delta T_{next} = \Delta T_{current} + 1sec$
$m\ undefined$	$\Delta T_{next} = 2\ \Delta T_{current}$

2.4.2 Non-linear method

For some measurement purposes, a large fraction of useful information is contained in a small fraction of flows [13], for instance, in traffic accounting, the main target is to catch *elephant flows* (*i.e.*, flows of large size). The content-independent sampling techniques (presented in Section 2.2.1), when applied to the aggregate traffic, lead to the packets belonging to the large flows most likely to be sampled, being therefore sufficient to achieve high accuracy for this purpose. However, this approach is not suitable for measurement tasks in which different sizes of flows may be of importance, such as network security applications, that require accurate estimation of *mice flows* (*i.e.*, flows of small size) [8], inasmuch as the flow-level traffic patterns often reveal anomalies in the network [39] [41] [51].

In this way, the technique proposed in [8] aims at increasing the accuracy of the overall traffic distribution by using an adaptive non-linear sampling method. The basic idea consists in classifying packets into flows during the selection process, adjusting dynamically the sampling frequency of different flows individually according to the counter value (flow size) of each flow. This is performed by individual an probabilistic sampling process running on every flow. Unlike techniques that try to predict the flow size in order to adjust the sampling frequency, *e.g.*, resorting to a linear auto-regressive prediction model or resorting to a linear prediction method, the usage of distinct functions for different flows allows to eliminate predictive functions.

For every flow identified, the MP maintains a counter c with the number of sampled packets from that flow. Then, each new packet belonging to the flow will be sampled following the function $P(c)$ that rules the sampling of each flow reducing the frequency with the increasing of c according to the equation:

$$P(c) = \frac{1}{f(c+1) - f(c)} \tag{2.3}$$

where $f(c)$, with $c \geq 0$, is a sampling function to be selected following the conditions:

- $f(c)$ is a real increasing convex function;
- $f(0) = 0$ and $f(1) = 1$;
- $f(c) < f(c+1) \geq bf(c) + 1$ with $b > 1$ and $c > 0$.

Although the results demonstrate a significant improvement in estimation accuracy, particularly for small-size flows, performing a different sampling frequency over different flows in parallel may lead to a considerable computational overhead.

2.4.3 Fuzzy logic method

Following the same principle that rules the adaptive method presented in Section 2.4.1, in which the interval between samples is reduced when higher activity is detected in the network and is increased when less activity is detected, the *fuzzy logic method* avoids attempting to predict the exact value of future traffic volume by adjusting the sampling frequency based on one or more premises and a single implication [28].

Unlike identifying network activity levels such as "high", "medium" and "low", this technique mimics the decision process employed by human brain [52], producing a sub-range of possible outputs, such as *change-slight*, *change-low* or *change-medium*. For this, a controller receives the current interval between samples ($\Delta T_{current}$) and the difference of traffic volume estimated between the last two samples (ΔX) as inputs. Then interprets the values into one or more fuzzy states for each input (as presented in Table 2.2). The mapping between numerical values from $\Delta T_{current}$ and ΔX to the fuzzy variables relies on previous experiences and may vary for different traffic scenarios, requiring therefore a training period in which the optimal spread and boundaries of the parameters could be ascertained.

The input states are then correlated yielding an output, also defined in terms of fuzzy variables (F_{out}), that must be further mapped to numerical values used to schedule the next sample event. As detailed in Table 2.3, two premises are correlated by using the

Table 2.2: *Fuzzy state inputs*

ΔX	$\Delta T_{current}$
No-Change (NC)	Small (S)
Change-Slight (CS)	Small-Medium (SM)
Change-Low (CL)	Medium (M)
Change-Medium (CM)	Medium-Large (ML)
Change-High (CH)	Large (L)

logic operator AND in order to provide a simple implication, that will be converted in a nonfuzzy output following Equation 2.4.

The mapping of the F_{out} fuzzy state into a numerical value follows a tree steps process using the correlation product method: (*i*) find the set of all rules that correspond to the input states by considering all permutation of states; (*ii*) choose the minimum degree value of the inputs among the matching rules; (*iii*) scale the shape of the function for the F_{out} value of each rule by the minimum input value defined in the previous step. This process is repeated for all the rules, the resulting scaled shapes of each function are combined and then their center of mass is calculated using Equation 2.4, in which z corresponds to the numerical value used for F_{out}, n is the number of rules combined, U_j and V_j are the peak of the shape for the combined functions for x-axis and y-axis, respectively [28].

$$z = \frac{\sum_{j=1}^{n}(U_j \times V_j)}{\sum_{j=1}^{n} V_j} \tag{2.4}$$

Despite the flexibility provided by the extensive sub-range of possible traffic scenarios, the experimental process of mapping these levels according to the type of traffic, and the necessity of storing them for future similar situations may lead to a excessive complexity of deployment.

2.4.4 Statistical method

This technique was introduced in [46] and further enhanced in [53] with the aim to optimize the sampling process considering the statistics of multiple parameters related to QoS metrics, *i.e.*, delay, jitter, throughput and packet loss ratio. Basically, the technique performs packet collection driven by the arrival time of the packet at the measurement point. Each period of collection is called section (a sample according the terminology presented in Section 2.1) and has a fixed size. Then the method adaptively adjusts the time interval between two consecutive sampling sections (called pre- and

Table 2.3: *Rules for the fuzzy mapping*

Rule	ΔX	$\Delta T_{current}$	F_{out}
1	No-Change (NC)	Small (S)	Increase-high (IH)
2	No-Change (NC)	Small-Medium (SM)	Increase-high (IH)
3	No-Change (NC)	Medium (M)	Increase-Low (IL)
4	No-Change (NC)	Medium-Large (ML)	Increase-Low (IL)
5	No-Change (NC)	Large (L)	No-Change (NC)
6	Change-Slight (CS)	Small (S)	Increase-high (IH)
7	Change-Slight (CS)	Small-Medium (SM)	Increase-Low (IL)
8	Change-Slight (CS)	Medium (M)	No-Change (NC)
9	Change-Slight (CS)	Medium-Large (ML)	Decrease-Low (DL)
10	Change-Slight (CS)	Large (L)	Decrease-Low (DL)
11	Change-Low (CL)	Small (S)	Increase-Low (IL)
12	Change-Low (CL)	Small-Medium (SM)	No-Change (NC)
13	Change-Low (CL)	Medium (M)	Decrease-Low (DL)
14	Change-Low (CL)	Medium-Large (ML)	Decrease-High (DH)
15	Change-Low (CL)	Large (L)	No-Change (NC)
16	Change-Medium (CM)	Small (S)	Decrease-Low (DL)
17	Change-Medium (CM)	Small-Medium (SM)	Decrease-Low (DL)
18	Change-Medium (CM)	Medium (M)	Decrease-High (DH)
19	Change-Medium (CM)	Medium-Large (ML)	Decrease-High (DH)
20	Change-Medium (CM)	Large (L)	Decrease-High (DH)
21	Change-High (CH)	Small (S)	Decrease-Low (DL)
22	Change-High (CH)	Small-Medium (SM)	Decrease-High (DH)
23	Change-High (CH)	Medium (M)	Decrease-High (DH)
24	Change-High (CH)	Medium-Large (ML)	Decrease-High (DH)
25	Change-High (CH)	Large (L)	Decrease-High (DH)

post-sampling sections) increasing the interval when the overall statistic of the traffic does not change for those two sections, and decreasing the interval when the overall statistic of the traffic for the two sections differ significantly. The quantification of statistical changes is defined as a threshold value specified by the user, that must be related to the parameter used to guide the method. The overall statistic (OS) is defined by the equation [46],

$$OS = |(\bar{m}_1 - \bar{m}_2)/\bar{m}_1| + |(md_1 - md_2)/md_1| + |(std_1 - std_2)/std_1| \qquad (2.5)$$

where,

- $\bar{m}_1$ *and* $\bar{m}_2$ are the mean values of a traffic parameter (*e.g.*, delay) being estimated for the pre- and post-sampling sections, respectively;

- md_1 *and* md_2 are the median values of the same parameter for the pre- and post-sampling sections, respectively;
- std_1 *and* std_2 are the standard deviation values of the pre- and post-sampling parameters, respectively.

The OS is then compared with the threshold defined by the user. If the OS value is less than the threshold, the interval between samples (ISI) is updated using,

$$ISI = ISI + round(ISI/\mu) \tag{2.6}$$

where μ is a predefined value that controls the update magnitude.

Otherwise, if the OS value is greater than or equal to the threshold value, then ISI is updated using,

$$ISI = ISI - round(ISI/\mu) \tag{2.7}$$

The advantages of this sampling method are easy implementation and fast response to the variation of the input traffic. The algorithm output however is to some extent sensitive to its initial parameter settings [46].

2.5 Packet sampling in practice

Beyond the vast number of new techniques, the diversification of communication services and their underlying requirements have also fostered a large number of research works focused on assessing the most suitable sampling technique for different measurement tasks. Several of these studies are mainly devoted to analyze and enhance the trade-off between sampling accuracy and overhead of the traditional techniques defined in [1], which essentially intend to minimize information loss while reducing the volume of collected data [13].

The notion that distinct sampling methods and their implementations represent an additional source of variability for interpretation of measurements is expressed in [25], where, after simulating various sampling approaches (also standardized in [1]) revealed that time-triggered techniques did not perform as well as packet-triggered when estimating the distribution of packet size and packet interarrival times. This occurs due to the burstiness in traffic patterns [54], in which the occurrence of many packets with small interarrival times leads time-based techniques to more misses than a count-based.

Nevertheless, comparing different count-based techniques (*i.e.*, systematic and random) the accuracy for the same parameters is very similar [2].

Furthermore, despite the probabilistic methods have been proposed aiming at improving the accuracy in estimating traffic statistics, as demonstrated in [55] and [56], they do not necessarily deliver such improvements facing deterministic approaches for a large set of metrics related to traffic classification and characterization. Moreover, for flow accounting, systematic sampling performs a little better [7] and for anomaly detection, random sampling causes both a significant decrease in detection performance and an increase in false negatives (*i.e.*, a loss of detectability) [41].

Unlike metrics oriented to the estimation of aggregated traffic volume, count-based techniques tend to produce more inaccurate estimations for flow oriented accounting when compared with time-based techniques. As two back-to-back packets of a flow in a stream will rarely be selected by spatial distributed sampling processes, usually only few flows get enough packets captured to achieve sufficient accuracy for accounting [7]. This also affects the performance of applications that require the analysis of consecutive packets, such as IDS [57], being therefore, time-based techniques more suitable for this type of measurements.

In addition, some works have demonstrated that conducting sampling only to a specific set of flows of interest can improve the estimation accuracy for tasks sensitive to flow sizes [58]. For instance, while traffic accounting targets mainly flows of large sizes [13], anomaly detection targets flows of small sizes [8].

When considering adaptive techniques, there is not a comprehensive comparative study in the literature. As these techniques are usually proposed aiming at enhancing the accuracy for a specific measurement goal, the comparative studies are generally limited to evaluate their improvements regarding a small number of schemes described in [1] (commonly the systematic count-based and random n-out-of-N). Typically, most of these works claim that their adaptive techniques outperform the traditional schemes within the scope in which they were proposed.

The lack of an encompassing study regarding the suitability of sampling techniques for multiple measurement tasks is not only limited by the usual specificity of adaptive techniques. It is also affected by the complexity of deploying them in current commercial sampling tools (that only support few approaches), hampering thus a wide adoption of features that make some techniques more efficient. This is evident in main sampling tools commercially available such as *Cisco Sampled Netflow* and *sFlow*.

Due to its large market share[1], *Cisco Sampled NetFlow* is one of the most used sampling tool, especially because it is embedded in many models of commercial routers and switches. The tool is composed by two modules, not necessarily available in all devices. The *deterministic NetFlow* module implements the systematic count-based technique and the *random sampled NetFlow* module implements the random n-out-of-N (both presented in Section 2.2.1). In this way, this sampling-based tool only provides count-based and content-independent techniques.

sFlow is an industry standard adopted by multi-vendor as the sampling technology embedded within their switches and routers. It was proposed as an RFC (*i.e.*, RFC 3176 [59]) that defines the sampling mechanisms implemented in an sFlow Agent for monitoring traffic, the sFlow MIB - *Management Information Base* for controlling the sFlow Agent, and the format of data used to forward sampled data to an external data collector [59]. As Cisco Sampled NetFlow, sFlow provides two count-based techniques, the systematic and random. However, the random technique works slightly different as it generates a random number for each incoming packet, compares the random number to a preset threshold and collects the packet whenever the random number is smaller than the threshold value.

Both tools allow the combination of sampling and filtering in order to improve measurement flexibility, for instance applying the sampling policy only to packets being forwarded through a specific interface. However, they do not provide crucial sampling strategies, such as time-based, relevant for activities related to anomaly detection, as discussed before. Similar limitations are also present in different measurement tools and network vendors, for instance *tcpdump*, *Alcatel cFlow*, *Juniper J-flow* and *Endace NICs*.

From the large coverage of related literature it is clear that the selection of a strategy for packet sampling depends on the type of measurement task in which it will be applied, therefore, there is not a single technique able to face all measurement goals. In this way, a measurement system able to accommodate different sampling strategies, whether standardized or not, aiming at providing high accuracy measurements while maintaining the computational overhead under control is not only an open issue, but also highly desirable.

[1]Source: IDC Worldwide Quarterly Server Tracker, 2014Q2, August 2014, Vendor Revenue Share for top vendors. Revenue is cumulative 4 quarters (Q3CY13 - Q2CY14).

2.6 Summary

This chapter has presented an encompassing overview of the current packet sampling techniques. It has included techniques standardized through IETF RFC 5475 [1] and techniques recently proposed, which frequently involve complex selection schemes, such as adaptive techniques. In addition, the chapter went further in identifying and discussing the applicability of sampling techniques in several measurement activities and their availability in current measurement tools. The literature review has shown that there is not a comprehensive comparative study addressing the vast number of sampling techniques regarding their performance in satisfying the various measurement objectives. However, it was possible to identify that, some of the less deployed techniques tend to achieve better performance than the classical ones for various traffic scenarios and measurement tasks. These observations sustain the need for analyzing the sampling techniques through its constituent parts, rather than a closed unit. This will allow to identify their common properties, which lead to the better performance, and address eventual constraints within a narrower and simpler scope. This topic will be covered in Chapter 4, after the presentation of a new *Multiadaptive* sampling technique in Chapter 3. The proposed technique is presented in a dedicated chapter due to be a contribution of this doctoral work.

Chapter 3

Multiadaptive Sampling Technique

As discussed previously, adaptive sampling techniques aim at increasing the measurement accuracy by driving packet selection through dynamic sampling policies targeting packets of interest for achieving a specific monitoring goal. In addition, some adaptive techniques also propose strategies to reduce the sampling overhead by reducing the volume of data processed, stored and transmitted.

The algorithms introduced in [28] and [46] (detailed in Section 2.4) basically adjust the sampling frequency of time-triggered sampling schemes by varying the interval between samples according to the activity level observed in the network. The main idea consists in reducing the interval between samples (increasing the sampling frequency) when high-level network activity is identified or conversely, increasing the interval between samples (reducing the sampling frequency) when low-level activity is identified. In this way, if the volume of traffic under measurement increases significantly, the volume of packets captured by the sampling technique will also increase. This may cause serious concerns, considering that sampling usually shares hardware resources with routing or switching processes.

In this context, this chapter presents a new multiadaptive sampling technique (MuST) [14] [15] [16] [17] based on linear prediction, aiming at reducing the amount of data involved in the network measurements without compromising the estimation accuracy. Therefore, the main objective is to reduce the measurement overhead and still assure that sampled traffic reflects the statistical characteristics of the global traffic under analysis. For this purpose, the traffic selection process considers the levels of network activity, being configured to reduce the measurement impact when the network activity increases or the measurement process tends to overload the measurement points. The multiadaptive behavior of the proposed technique is achieved considering both the interval between samples and the sample size as adaptive parameters, bounded by proper thresholds to guarantee the representativeness of samples in capturing the network behavior. The resulting gain achieved by MuST is discussed in Chapter 7.

3.1 Design goals

As mentioned, the multiadaptive sampling technique proposed aims at improving the sampling reactivity and efficiency, considering both the interval between samples and the sample size as adjustable parameters [15]. In this way, the present proposal pursues the following design goals:

(i) the adaptive nature of the technique should be driven by simplicity of implementation and low consumption of resources;

(ii) the adaptive sampling process should be defined in order to minimize the impact of sampling on the normal network operation while keeping high-accuracy levels. Therefore, the technique should gauge the past and current network activity in order to estimate adequate parameters to guide the traffic selection process.

The first goal motivates the adoption of a sampling approach based on linear prediction, as proposed in [28]. In fact, as discussed in Section 2.4, the use of linear prediction leads to lighter solutions when compared to fuzzy logic adaptive approaches. However, in the adaptive process described in [28], the underlying processing overhead is still significant as the whole network traffic is considered for the definition of the reference parameter, regardless of packets belonging or not to a sample. In MuST, the technique described in [28] is modified so that only packets belonging to previously collected samples are taken into account to drive future sampling decisions, clearly reducing the processing overhead.

To pursue the second goal, an adaptive sampling process is defined to react autonomously and self-adapt to distinct network loads and traffic characteristics. In this process, the sampling frequency is increased or decreased whenever there is a noticeable increase or decrease in the network activity in order to allow detecting new traffic patterns. As increasing the sampling frequency implies higher consumption of resources (processing and storage), the sample size should be consequently reduced to mitigate the overhead increase. To guarantee that a representative amount of data is obtained toward capturing the network behavior, the adaptive parameters need to be properly bounded by thresholds.

The sampling frequency can be adjusted varying the interval between samples, resorting to a linear prediction function for adapting the sampling frequency based on [28]. The new predictive function is presented in Equation 3.1, whereas the set of rules determining the change factor in the interval between samples are presented in Equation 3.2 and Table 3.1. This formulation is detailed in the following section.

As none of the techniques available so far present features for adjusting dynamically the sample size, this work proposes a set of rules for varying the sample size according to the level of network activity. These rules increase and decrease the sample size considering the observed network behavior. The dynamic variation in sample size complies with the criteria presented in Table 3.2. This will also be detailed below.

3.2 Multiadaptive sampling technique description

MuST takes into account the last N samples to estimate the future value of the reference parameter, which is then used to determine the next interval between samples and the size of the next sample. Thus, for a sampler of order N, the expected value X_p of the reference parameter for the next collected sample is defined as

$$X_p = X_N + \frac{\Delta T_N}{N-1} \sum_{i=1}^{N-1} \left(\left| \frac{X_{i+1} - X_i}{\Delta T_i} \right| \right). \tag{3.1}$$

In Equation 3.1, the variable X represents the values of the reference parameter of the last N samples, being X_N the value of the most recent sample. A second variable T represents the intervals between samples, where each ΔT_i is the time elapsed between the end of the sample X_i and the beginning of the sample X_{i+1}, i.e., $\Delta T_i = T_{i+1} - T_i$ for all $1 \leq i \leq N$ and $N > 1$.

3.2.1 Defining the interval between samples

When a new sample is collected, the corresponding value of the reference parameter S is compared with the expected value X_p in order to determine a factor of change m. Depending on the value of m, a set of rules is applied to define the sampling interval ΔT_{N+1}, which will determine the start of the next sample. The factor m, obtained comparing X_p and S, is given by

$$m = \begin{cases} \frac{X_p}{S} & if S \neq 0; \\ 1 & otherwise. \end{cases} \tag{3.2}$$

The fraction in Equation 3.2 returns a value close to 1 when the expected value X_p is close to the current value of S, corresponding to a correct estimate. In this case, the range of values for m is defined as varying between $m_{min} = 1 - \sigma$ and $m_{max} = 1 + \sigma$, *i.e.*,

$$1 - \sigma < m < 1 + \sigma$$

where σ allows to adjust the degree of adaptiveness (or reactivity) in the estimation process. As discussed in [28], considering a 10% variation in the reference parameter (representing a variation in the network activity) has led to an adequate regulation in presence of multiple traffic types. Therefore, these values are set as $m_{min} = 0.9$ and $m_{max} = 1.1$.

If $m < m_{min}$ the predicted value of the reference parameter was underestimated, indicating more network activity than expected. Thus, the interval between samples is decreased according to m variation to achieve more accurate values in the following predictions.

On the other hand, if $m > m_{max}$ the value of the reference parameter was overestimated in the prediction, and the network activity is slowing down. In this case, the interval between samples is exponentially increased in order to converge faster to its maximum value, reducing the measurement overhead.

If the value of S is null, representing that no traffic has been captured, e.g., due to a reduced network load or a temporary link failure, m assumes a unitary value, which allows to keep the adaptive sampling parameters stable[1]. In this case, the current ΔT_N is assumed as the next interval between samples.

Table 3.1 lists the rules used to generate the next sampling interval ΔT_{N+1}.

An additional threshold is defined to prevent ΔT from increasing indefinitely, thus

[1]Note that, X_p keeps incorporating S, becoming null after N samples with $S = 0$. This behavior is adequate to resume properly the adaptive process when new traffic is detected. In practice, network links tend to exhibit load as, at least, control traffic is crossing the links. Therefore, successive iterations with $S = 0$ are likely due to link failure, which would be detected and handled at a higher network management layer.

Table 3.1: *Rules to define the next interval between samples*

current m	next ΔT
$m < m_{min}$	$\Delta T_{N+1} = m\ \Delta T_N$
$m_{min} \leq m \leq m_{max}$	$\Delta T_{N+1} = \Delta T_N$
$m > m_{max}$	$\Delta T_{N+1} = 2\ \Delta T_N$

guaranteeing a minimum number of samples to obtain representative data for new predictions. Similarly, the maximum frequency of sampling is also limited so that the sampling interval does not tend to zero, which would result in capturing all traffic. These limits should weight and be adjusted according to the existing link capacity. In the present study, similarly to [28], a minimum and a maximum interval between samples of 0.1s and 8s, respectively, has shown adequate based on experimental tests under diverse traffic scenarios.

Figure 3.1 illustrates the evolution of the interval between samples as a function of a linear variation of the m factor. As shown, the reactivity is smoother when the network activity increases, i.e., the next ΔT decreases proportionally to m when $m < m_{min}$. Conversely, the reactivity is higher in presence of low network activity, i.e., the time interval between samples varies exponentially when $m > m_{max}$.

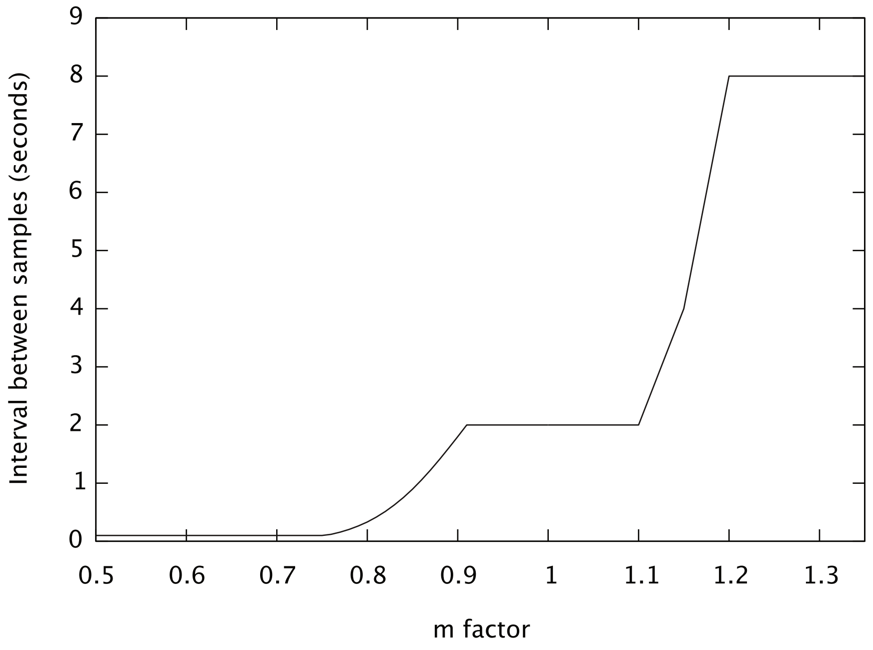


Figure 3.1: *Evolution of interval between samples according to m variation*

3.2.2 Defining the sample size

For adapting the sample size, the factor m is also considered as an indicator of network activity. Table 3.2 presents the rules used to define the next sample size, where ΔS_N represents the current sample size and ΔS_{N+1} the size of the next sample to be collected.

According to Table 3.2, in moments of increased activity, the sample size is decreased proportionally to m. This reduction in sample size, associated with the higher frequency in the sampling process, aims at reducing the overhead at measurement points. In presence of less network activity, the sample size is adjusted by a factor[2] k, with $k = 0.15$ (see rational in Section 3.3.1). This allows to collect more data about the network in less critical periods of its operation, in sparse sampling events.

Table 3.2: *Rules to define the next sample size*

current m	next ΔS
$m < m_{min}$	$\Delta S_{N+1} = m\ \Delta S_N$
$m_{min} \leq m \leq m_{max}$	$\Delta S_{N+1} = \Delta S_N$
$m > m_{max}$	$\Delta S_{N+1} = \Delta S_N + (k\ \Delta S_N)$

Similarly to the definition of the time interval between samples, the variation of sample sizes is also bounded. The imposed thresholds avoid small samples, which make difficult estimating parameters statistically, as well as samples excessively large, closely matching a total traffic capture. These limits also depend on the existing links capacity, being here considered a minimum and maximum sample size of 0.1s and 2s, respectively. Figure 3.2 shows the evolution of the sample size for a linear variation of the m factor.

For completeness, the pseudocode of MuST algorithm representing the overall sampling operation is included in Appendix A. In addition, the flowchart presented in Figure 3.3 describes sequentially the multiadaptive operation.

3.3 Assessing variable parameters

Although the overall performance analysis of MuST is covered in Chapter 7, this section presents experiments aiming at assessing the impact of tuning the factor k when adapting the sample size and the order of prediction N. These studies are performed

[2]The parameter k was firstly introduced and experimentally tested in [46] with the aim of defining the variation in the time interval between samples based on a set of comparative statistics between adjacent samples. In the present proposal, the parameter k is considered as the changing factor in the sample size, being defined to force an overhead reduction.

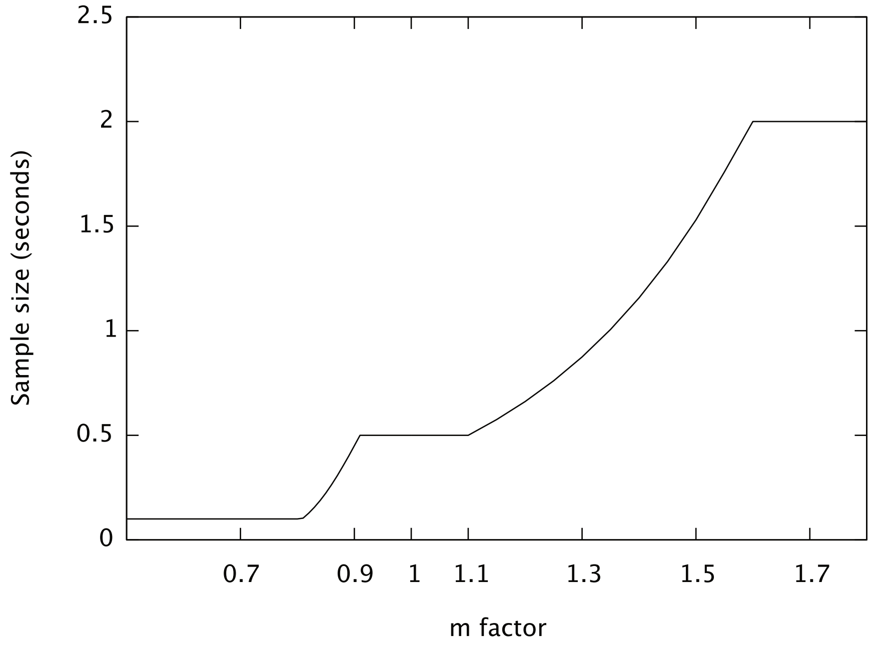


Figure 3.2: *Evolution of sample size according to m variation*

evaluating the MuST performance for two distinct scenarios: one typically more regular (OC-48) and other with higher variability (SIGCOMM08). For each traffic type, the analysis is focused on the cumulative volume of sampled data (in Mbytes), RME (Relative Mean Error) of the estimated mean throughput and number of samples, for different values of k and N.

Table 3.3 illustrates the characteristics of the real traffic traces used in the tests. These traces were selected to provide a representative range of traffic scenarios to assess the versatility of MuST.

Table 3.3: *Traffic Scenarios*

Traffic Label	Characteristics	Duration	Available
SIGCOMM08	Captured during the SIGCOMM 2008 conference, including all the participants communications through IEEE 802.11a access points.	8 hours	CRAWDAD [60]
OC-48	Captured passively in a backbone link OC48 in a large ISP from the US West Coast.	5 minutes	CAIDA [61]

3.3.1 Impact of k

Regarding the impact of k when adapting the sample size, as the results illustrate (see Figure 3.4), a $k = 0.10$ leads to the lowest volume in sampled data, however, it also

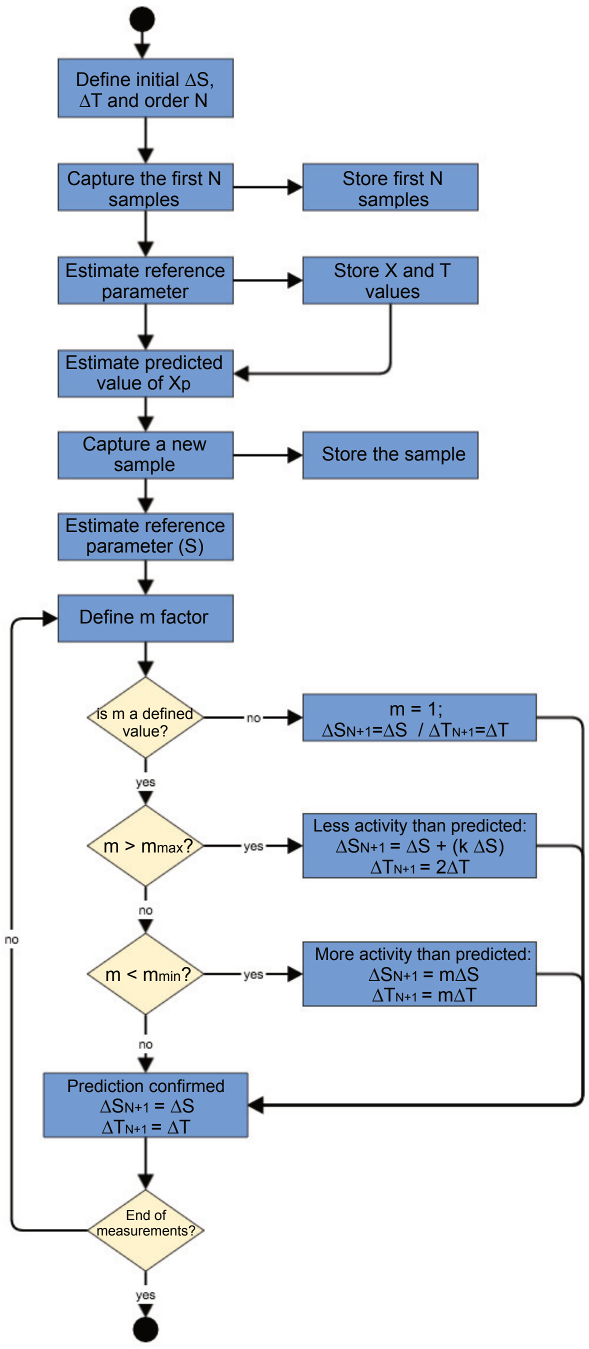
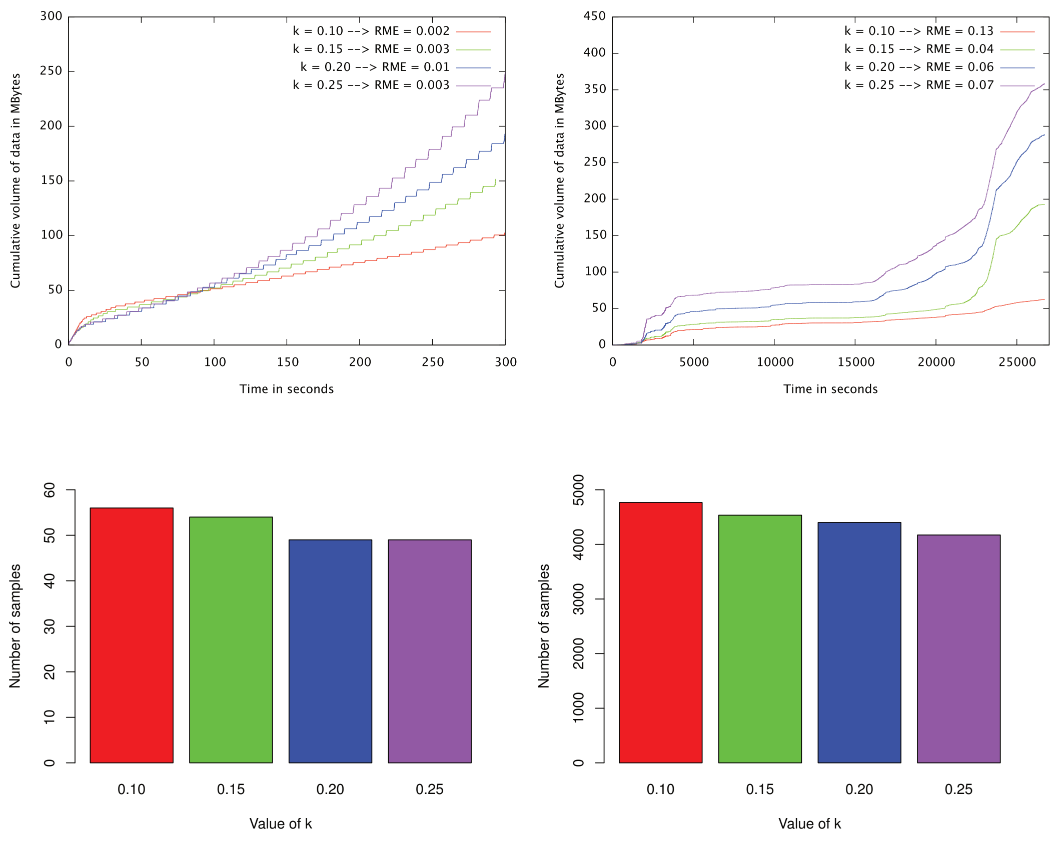

Define initial ΔS, ΔT and order N
Capture the first N samples
Store first N samples
Estimate reference parameter
Store X and T values
Estimate predicted value of Xp
Capture a new sample
Store the sample
Estimate reference parameter (S)
Define m factor
is m a defined value?
no
m = 1; ΔSN+1=ΔS / ΔTN+1=ΔT
yes
m > mmax?
yes
Less activity than predicted: ΔSN+1 = ΔS + (k ΔS) ΔTN+1 = 2ΔT
no
m < mmin?
yes
More activity than predicted: ΔSN+1 = mΔS ΔTN+1 = mΔT
no
Prediction confirmed ΔSN+1 = ΔS ΔTN+1 = ΔT
End of measurements?
no
yes

Figure 3.3: *MuST - Operational flowchat*

causes an increase in RME (one order of magnitude) in the presence of more variable traffic. As shown in Figure 3.4 (b), with $k = 0.10$, the adaptive behavior of MuST is less sensitive in capturing traffic variation. The results also show that a value of $k = 0.15$ presents a good compromise between the variables under study (data volume and RME) for both traffic types, although it does not lead to the smallest number of samples for the values of k considered. The obtained differences for the number of samples are, however, not significant, meaning that MuST is well-behaved for $k = 0.15$.



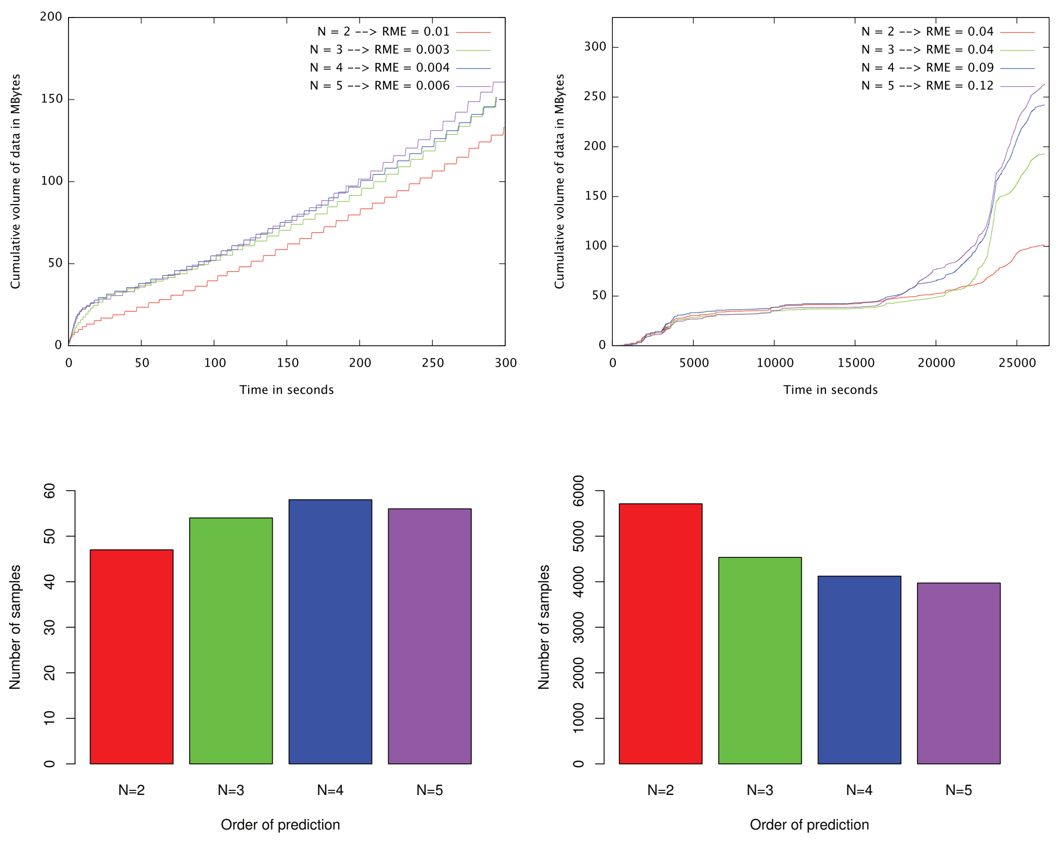
(a) OC-48 (b) SIGCOMM08

Figure 3.4: *Impact of k on MuST performance: Cumulative data volume (up); Number of samples (down)*

3.3.2 Impact of the order of prediction

Similarly to the study of k presented above, the following experiments aim at evaluating the impact of varying the order of prediction N on MuST performance. Figure 3.5 presents the results comparing the overhead and RME for OC-48 and SIGCOMM08 traces for distinct values of N. Notice that higher values of N correspond to configure the adaptive traffic selection process with more information of past samples, i.e., including more past memory. Generally, a larger past memory leads to less reactive mechanisms, and shorter past memory improves the reactivity to short term traffic fluctuations. The degree of this reactivity may affect the stability of sampling mechanisms.

According to the obtained results, an increase in N conducts to an overhead increase



(a) OC-48 (b) SIGCOMM08

Figure 3.5: *Impact of N on MuST performance: Cumulative data volume (up); Number of samples (down)*

regarding the volume of data collected, for all traffic types. However, for $N = 5$, it is clear that collecting more data does not lead to higher accuracy in the throughput estimation (see RME); this is particularly visible in Figure 3.5 (b). This means that lower reactivity also reduces the ability to correctly measure bursty traffic. Nevertheless, when taking $N = 2$, corresponding to the smallest overhead regarding the amount of sampled data, it does not imply a gain in accuracy, as visible for OC-48 traffic. Considering the number of samples, varying N does not lead to a linear distribution for all traffic types. This is due to each particular traffic characteristics, which is inline with the adaptive behavior of MuST when facing traffic fluctuations, described in Section 3.1.

Globally, the values obtained for RME are very low for all traffic scenarios. Therefore, depending on the objective or usefulness of throughput estimation, the value of N can be tuned to achieve an optimal compromise between overhead and efficiency.

3.4 Summary

This chapter has presented a new multiadaptive sampling technique (MuST) aiming at reducing the sampling overhead in network measurements. The technique relies in a controller based on linear prediction, that drives the packet selection considering the levels of network activity, being configured to reduce the measurement impact when the network activity increases and the measurement process tends to overload the MPs. The multiadaptive behavior of the proposed technique is achieved considering both the interval between samples and the sample size as adaptive parameters, bounded by proper thresholds to guarantee the measurement representativeness of samples. In addition, the operational parameters that rule the algorithm reactivity have been evaluated regarding the overhead in the amount of data resulting from the sampling process for real traffic scenarios. An extensive analysis of MuST performance in comparison with different sampling techniques in diverse traffic scenarios and measurement activities is presented in Chapter 7. Next chapter presents a taxonomy of sampling techniques able to drive the modular deployment of efficient sampling strategies.

Chapter 4

Taxonomy of Packet Sampling Techniques

As previously discussed, an effective sampling-based measurement system must support distinct sampling techniques, inasmuch as the type of traffic and the measurement goal to fulfill may affect directly the decision on the sampling technique to be used.

Although the RFC 5475 [1] provides insights into the deployment of complex sampling techniques by the composition of different blocks of filtering and sampling schemes, some techniques may become too complex, requiring the specification of an *ad hoc* description and its addition as a new scheme to the information model. The use of non consolidate models may, however, introduce significant challenges, hampering the large adoption of more appropriate techniques in measurement systems.

A large coverage of the related literature shows that most of the sampling techniques, whether simple or complex, share a set of structural components, based on standard schemes, or as new strategies, arranged orthogonally with classical schemes or completely disjunct. In this way, describing these components in a modular and hierarchical structure able to foster a flexible and simple deployment of a comprehensive number of techniques represents a key role toward effective measurement systems based on sampling. This will allow exploiting specific features suitable for reaching better accuracy in each measurement purpose and traffic scenario.

With the aim to provide a common ground for current and forthcoming sampling proposals, this chapter presents a taxonomy of sampling techniques [18]. This involves identifying and classifying functional blocks of previously presented techniques in order to support the design of sampling strategies adjusted to different measurement scenarios. In the classification criterion, a set of features related to sampling *granularity*, *selection scheme* and *selection trigger* are identified and proposed as the components distinguishing current solutions. The main contribution of this taxonomy is the ability to describe most of the current sampling techniques though the composition of, at least, one approach from each component in a modular structure. This aims at fostering the deployment and wide adoption of both classical and new sampling techniques within a complete architecture of measurement systems, presented in Chapter 5.

4.1 Taxonomy proposal overview

The defined taxonomy fragments the sampling techniques 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 both classic and recently proposed sampling techniques. An overview of the taxonomy is illustrated in Figure 4.1.

- *Granularity* - identifies the atomicity of the element under analysis in the sampling process: in a flow-level approach, the sampling process is only applied to packets belonging to a flow or to a set of flows of interest; in a packet-level approach, packets are eligible as single independent entities;

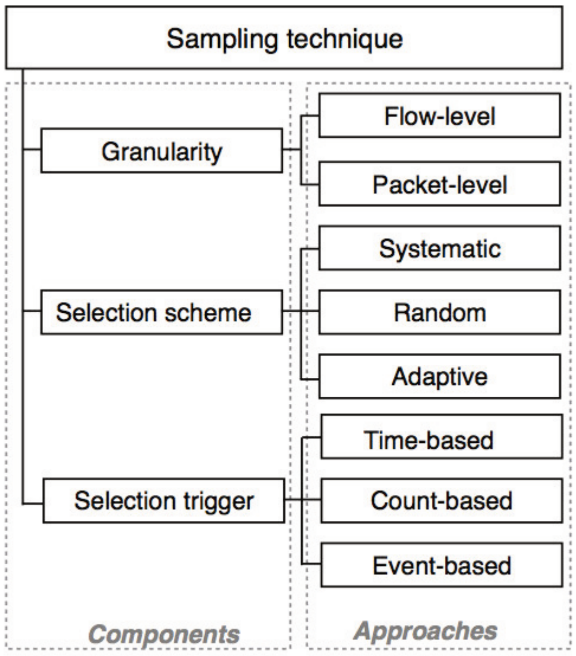


Figure 4.1: *High-level view of sampling taxonomy*

- *Selection scheme* - identifies the function defining which traffic packets will be selected and collected; this scheme may follow a deterministic, a random or an adaptive function;
- *Selection trigger* - determines the spatial and temporal sample boundaries; it may use a time-based approach, a count-based approach or an event-based approach.

These components are further detailed in the upcoming sections.

4.2 Granularity

This component identifies which segment of traffic is considered in the sampling process and in the data reporting format. During the selection of packets, the sampler may consider all traffic traversing a MP or just part of it, targeting specific flows of interest. Generally, this decision depends on the network task (or measurement objective to fulfill), the network parameters being monitored, and the available communication and computational resources.

4.2.1 Flow-level sampling

According to RFC3697 [62], a flow is defined as a stream of packets sent by a particular source to a unicast, anycast or multicast destination, which exhibits specific properties or attributes in common. Traditionally, these properties (also called a flow key) are identified based on five fields (5-tuple) of the packet header, namely source and destination IP addresses, source and destination ports, and type of protocol. In addition, RFC 2724 [63] and RFC 7011 [64] extend flow identification based on application layer information, MPLS (Multiprotocol Label Switching) labels or fields derived from packet treatment (e.g., next-hop IP address, etc.).

In terms of traffic sampling, the flow-level approach consists in applying the traffic capture policy only to packets belonging to a flow or a set of flows of interest. This involves classifying packets into flows before or during the sampling process [2]. Although considering a subset of flows may reduce the volume of data captured, stored and transmitted by the MP, this approach may increase the computational weight insofar all incoming packets must be processed to identify which flow they belong to. It also requires prior knowledge of which flows should be measured or some strategy to decide automatically which flows should be sampled.

Note that *flow-level sampling* is different from *flow sampling* (discussed in RFC 7014 [65]), that consists in capturing all packets that belong to a particular flow. Nevertheless, the strategies used in selecting flows, presented in [65] (*i.e.*, *systematic* and *random*), also may be applied to flow-level sampling context. In addition, different strategies also can target specific flows according to the measurement purpose. An example is the method introduced in [66], called *smart sampling*, that addresses the correct estimation of flow size distribution. These strategies are presented below already considering the necessary adaptations toward flow-level sampling:

Deterministic flow selection resorts to a deterministic function in which a set of flows are selected according to the number of flows arriving the MP or to a time interval of observation. In the first case, the MP selects every $N-th$ arriving flow to be considered for sampling, independently of the traffic type. The selection of a flow is then based on the first packet of a flow, in which every time a packet belonging to a new flow arrives at the MP, a counter is increased. If the counter is increased to a multiple of N, this flow will be considered for packet sampling. In the second approach, the packets from every flow observed at the MP between a time-based interval are elected to participate in the sampling process [65].

Random flow selection is based on a random process to select flows following a *n-out-of-N* or a *probabilistic* scheme. In *n-out-of-N*, n flows are selected out of the parent population, which consists of N incoming flows. It may involves generating n different random numbers in the range [1,N] and then selecting all flows with arrival position equal to one of the random numbers. In *probabilistic* flow selection, the decision of whether or not a flow is selected to participate in packet sampling follows a predefined probability that may be uniform (*i.e.*, with the same selection probability for all flows) or non-uniform (*i.e.*, where the selection probability can vary for different flows). The probabilistic scheme implies that the number of selected flows can vary [65].

Smart sampling aims to enhance the ability to identify accurately the distribution of the traffic by controlling the flows that will participate in the sampling process according to a threshold previously defined. For every incoming flow, the MP maintains a counter with the size of the flow (in bytes or number of packets), then flows of size greater than the threshold are always selected, while smaller flows are selected with a probability proportional to their size [66].

As discussed in Section 2.5, the flow-level sampling approach enhances the estimation accuracy for metrics related to tasks such as flow accounting and characterization.

In addition, as a group of packets share the same features (*i.e.*, flow key), it is possible to aggregate flows and thereby reduce the storage and transmission requirements to a manageable amount. In this way, flow-level sampling is expected to comply with IPFIX - *IP Flow Information Export* (RFC5470 [67], RFC6183 [68]), a protocol that defines an architecture for exporting the measured IP flow information from a MP (detailed in Section 5.4.3).

4.2.2 Packet-level sampling

In this approach, incoming packets to the MP are considered single independent entities. Conversely to flow-level sampling, at packet-level the packets do not need to be previously classified into flows, which may reduce drastically the computational requirements of processing every packet. Furthermore, collecting packets indistinctly turns packet-level sampling into a flexible and appropriate solution to be used in general purpose measurement tasks and aggregated estimations, in presence of diverse traffic types.

On the other hand, packet-level sampling may be difficult to deploy over large and high-speed networks, due to the challenges regarding the storage and transmission of measured data in environments with many flows.

As regards the exporting of sampled packets, a protocol based on IPFIX informational model (RFC5102 [69]), modified to report on single packets rather than on flows, is presented in RFC5474 [70]. This protocol defines mandatory contents for basic reports and an extended version able to include all fields required in RFC5102 [69]. These protocols are detailed in Section 5.4.3.

A strategy used by *Cisco Sampled Netflow* (presented in Section 2.5) aiming to cope with data volume exporting issues consists in collecting packets indistinctly in a first instance, for subsequent filtering or aggregation. Note that, although Netflow and Sampled Netflow are flow-based tools (state information is maintained per flow), in fact, the sampling selection scheme is packet-based, justifying its classification in this approach.

4.3 Selection scheme

The selection scheme identifies the selection function that determines the pattern under which packets will be selected and collected. This scheme can follow a *systematic*, a *random* or *adaptive* function.

4.3.1 Systematic sampling

In systematic sampling, the process of packet selection is ruled by a deterministic function which imposes a fixed sampling frequency, independently of the packet contents or treatment. In this scheme, only equally spaced traffic portions are collected, *i.e.*, sampling triggers are periodic (see Section 4.4). The systematic count-based (used in Cisco Sampled NetFlow) and systematic time-based techniques, introduced in [1] and detailed in Section 2.2.1, are examples of this approach.

This approach is usually simple to develop and deploy, however, as discussed in Chapter 2, there is an inherent risk of obtaining biased samples if the packets being sampled exhibit a periodic structure which is related to the deterministic function. This may lead to inaccurate measurement results due to the deterministic behavior of sampling facing the variable nature of network traffic. Another potential drawback is that systematic sampling is to some extent predictable and, hence, open to deliberate manipulation [2].

4.3.2 Random Sampling

The random selection scheme aims to avoid biasing samples by ruling the sampling frequency through a random process, usually resorting to a pseudorandom generator or to a probabilistic function.

The pseudorandom approach tries to avoid predictability choosing values exponentially distributed (for time-based trigger, discussed in Section 4.4.1) or geometrically distributed (for count-based trigger, discussed in Section 4.4.2) [2]. A common strategy following this principle is performed by inducing the random function generator to converge to a required sampling rate, ensuring that the sampling frequency distribution is limited by maximum and minimum values. The *n-out-of-N* technique presented in [1] and used in Cisco Sampled NetFlow follows this approach, where n packets are randomly selected from a traffic population of N packets, generating numbers in the range $[1, N]$ and then selecting all packets that have the corresponding packet position. This technique is detailed in Section 2.2.1.

As regards the probabilistic approach, the decision about the sampling frequency follows a predefined probability density function. The probabilistic function can be uniform, where all packets have an equal probability to be selected, or non-uniform, where the packets have different probability of selection. For instance, in [12] it is introduced a sampling technique based on a probabilistic scheme for anomaly detection, namely network scans, SYN flooding and worms. This technique divides time into strata and then selects an incoming packet with a probability, which is a decreasing function f of the predicted size of the flow the packet belongs to.

Random approaches are also difficult to deploy for estimating multipoint metrics such as end-to-end delay, even in flow-level approaches, as the sampling processes running in MPs involved are not correlated, and there are no guarantees that samples will be composed by the same packets [49].

4.3.3 Adaptive sampling

In this approach, the sampling technique is endowed with the ability to change the selection of packets during the course of measurements. This flexibility aims to identify the most important parts of a traffic stream according to the measurement needs or to save network resources during critical periods of its operation.

Adaptive packet sampling usually resorts to controllers, that consist in modules able to sense the network status by observing some parameter suitable to represent the traffic dynamics. These controllers may be based on linear prediction, fuzzy logic or other particular adaptive strategies and mechanisms (as detailed in Section 2.4) that consider the traffic behavior, the packet content or the network status to rule sampling pattern changes.

As discussed in Section 7.3, although the use of an adaptive strategy may suggest higher consumption of resources, some adaptive techniques may introduce less computational weight, regarding CPU load and memory consumption, even when compared with classical techniques [21].

4.4 Selection trigger

In network measurements based on sampling, only a subset of all packets traversing MPs is selected and considered to estimate network metrics. To achieve this, a trigger is defined to determine the start and the end of a sample, and consequently the interval between samples. In this way, a selection trigger is classified as *time-based*, *count-based* or *event-based* as described below.

4.4.1 Time-based

A time-based approach defines that the beginning and the end of a sample is determined based on packet arrival timestamping. Its deployment consists of using a first countdown timer within which all packets arriving at the MP are selected for the sample and a second countdown timer within which all incoming packets are ignored for measurement purposes.

Considering the example in Section 2.2.1 (*i.e.*, Figure 2.4), when the trigger fires the beginning of a new sample, the MP waits for the first bit of the next incoming packet and starts the collection. When the trigger fires the end of sampling, the MP continues the collection until the last bit of the current packet and then interrupts the selection process.

This may be combined with different approaches from granularity and selection scheme components in order to compose different sampling techniques. For instance the systematic count-based (presented in Section 2.2.1) resorts to a *systematic* selection scheme in which the sample size and the interval between samples are set at the beginning of the sampling process and remain invariant until the end. Another example is the multiadaptive technique presented in Chapter 3, in which is proposed an *adaptive* technique able to adjust both the sample size and interval between samples according to the network activity aiming at reducing the computational requirements involved in sampling processes [14].

In terms of performance, as discussed in Section 2.5, some studies have demonstrated that time-based triggers are less robust than count-based (see Section 4.4.2) when applied to traffic characterization [25], being affected by the bursty nature of network traffic. However, they may be suitable for applications that require the analysis of consecutive packets, such as IDS [57].

Traffic burstiness may also affect the volume of data involved in a time-based sampling process, and hamper the adoption of strategies which define the optimal number of samples beforehand (as it is difficult to anticipate the number of packets arriving at the MP in each sampling interval). As discussed in Chapter 7, the amount of data collected in time-based techniques is often higher than in count-based ones. However, as there is no significant computational activity during the interval between samples, such as packet counter increments, this approach may achieve a significant reduction in the ratio of CPU load and memory usage per MByte collected and stored [21].

4.4.2 Count-based

The count-based approach defines that the beginning and the end of a sample are driven by the spatial position of the packet within the traffic stream, using counters which are independent of the packet arrival timestamp.

An example of using this approach is presented in Figure 4.2, where a *systematic flow-level sampling* with sample size equal to 1 and interval between samples equal to 4 (*i.e.*, 1 packet out of every 5 incoming packets) is applied only to packets belonging to flow f_1. For this, the counter is decremented for each packet from flow f_1 arriving at the MP; when the counter reaches zero, a new sample starts, in this case collecting only one packet.

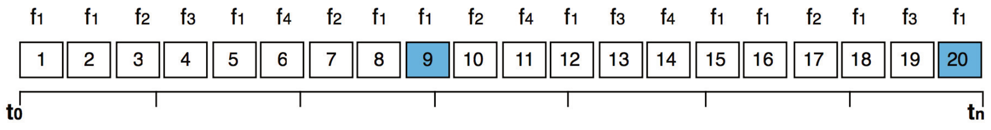


Figure 4.2: *Count-based*

Considering measurements of aggregate traffic, count-based techniques allow anticipating which proportion of the traffic will be collected and stored. Thus, this approach is suitable for environments with limited resources, or for applications where it is necessary to determine the optimal number of samples for a specific accuracy, as discussed in [71]. In this approach, every packet arriving at the MP must be processed to shift the packet counter, therefore, the computational weight involved is directly related to the total number of packets in the traffic under analysis [21].

4.4.3 Event-based

In this approach 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 MP or a more complex observation. The content-dependent schemes [1], detailed in Section 2.2.2, are examples of sampling techniques classified as event-based.

The packet contents corresponds to the union of the packet header (which includes link layer, network layer, and other encapsulation headers) and packet payload [1]. Sampling techniques based on this strategy may predefine some values of the packet header and then, all packets in which these values match are selected for the sample. This approach is usually called *property match filtering* [1].

Hash-based techniques (presented in Section 2.2.2) are also considered event-based, once the hash function is applied to the packet contents and then the packet is selected if the hash value falls within a selection range. This approach is sometimes used to emulate random sampling by selecting a proper range of hash values [1].

Although event-based sampling allows collecting a specific range of packets of interest, as it involves processing all incoming packets to identify the event, it may increase the overhead in the equipment.

4.5 Hybrid techniques

There are several techniques that combine approaches of the same taxonomy component, usually from the selection trigger or selection scheme. These techniques aim at enhancing traffic sampling, although the overlap increases the computational cost for traffic measurement.

An example of a hybrid technique is the use of an event-based trigger that fires upon observation of a packet with specified contents, after which any incoming packets within the next t seconds are selected to compose the sample, using a *time-based* approach [2]. This solution also may be deployed in conjunction with a *count-based* approach by capturing the first n packets arriving at the MP after identifying the event. As exemplified in Figure 4.3, the event is the first packet of a new flow observed in the stream and the sample size is equal to 3. This strategy may be of interest for traffic classification or security tasks.

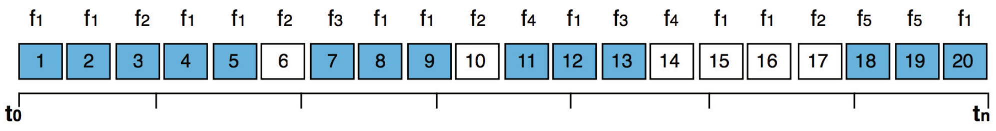


Figure 4.3: *Hybrid technique - event-based and count-based*

4.6 Sampling techniques composition

Following the taxonomy presented above, it is easy to classify most of classical and recent sampling techniques as well as to ground the definition of future proposals. Figure 4.4 exemplifies how the components defined in the taxonomy might be organized to deploy a sampling technique. The resulting sampling structure can be linear or hybrid (detailed in Section 4.5) depending on how sampling approaches are elected per component.

Figure 4.4 (A) corresponds to a technique defined in [1] available in most deployed sampling tools, *e.g.*, Cisco NetFlow and sFlow. Although the technique represented in Figure 4.4 (B) is also defined in [1], it is scarcely deployed in the current network measurement panorama, even considering its importance for IDS [57]. The technique represented in Figure 4.4 (C) illustrates the flexibility in deploying new sampling profiles. This technique might be used for monitoring a specific service and adapting the sampling frequency in response to some event observed into its own traffic. Some techniques able to be used in this context are presented in [13] [45].

Details about the relationship among the taxonomy components and their interactions are discussed in Chapter 6.

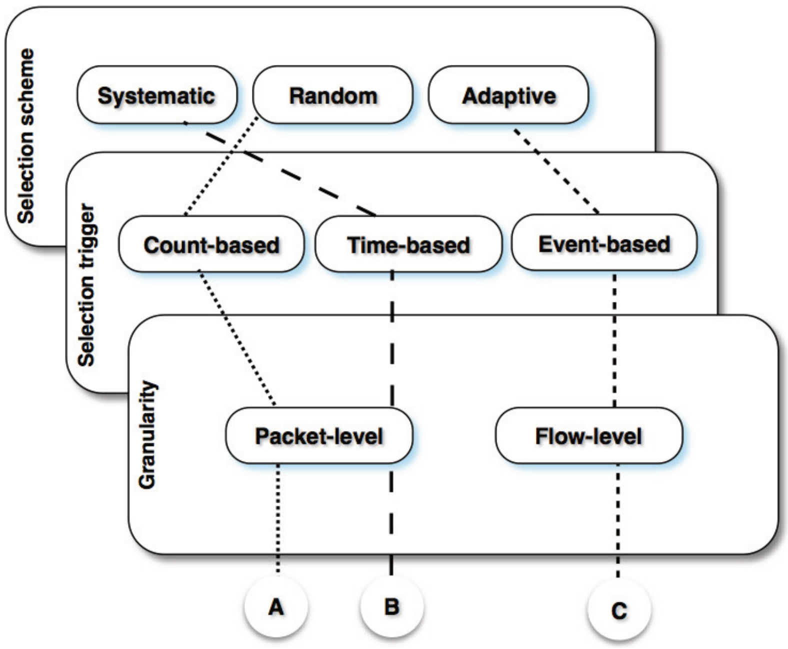


Figure 4.4: *Example of sampling technique composition*

4.7 Comparative summary of sampling techniques

Table 4.1 presents a summary of the most used and referenced sampling techniques classified according to the proposed taxonomy. This comparative study, along with insights throughout this work, allows a clearer positioning of existing sampling proposals, being both a research contribution in the area and a useful road map for deciding on the most suitable sampling technique to use.

Table 4.1: *Taxonomy of sampling techniques*

Sampling Proposals	**Granularity**		**Selection scheme**			**Selection trigger**		
	Pkt	Flow	Sys	Rnd	Adp	Cnt	Time	Event
Adaptive linear prediction [28]	√				√		√	
Adaptive non-linear sampling [8]		√			√	√		
Adaptive random sampling [72]	√			√		√		
Adaptive statistical sampling [46]	√				√		√	
Botnet-aware adaptive sampling [37]	√				√	√		
Distributed adaptive sampling [45]		√			√			√
Flow statistics trivial sampling [47]	√		√	√		√		
Fuzzy regulator adapt. sampling [73]	√				√		√	
Hash-based sampling [1]	√		√					√
Systematic count-based [1]	√		√			√		
Systematic SYN sampling [6]	√		√			√		√
Systematic time-based [1]	√		√				√	
Modified FLC sampling [74]	√				√		√	
Multiadaptive sampling [14]	√				√		√	
Opportunistic sampling [13]		√			√			√
Random n-out-of-N [1]	√			√		√		
Random sampled NetFlow [48]	√			√		√		
Resource conserving sampling [9]		√			√	√		
Sample and hold [30]	√			√		√		
Sampled NetFlow [48]	√		√			√		
sFlow [59]		√		√		√		

4.8 Summary

Grounded by a comprehensive review of the classic and new strategies of packet sampling, this chapter has defined a taxonomy of sampling techniques with the aim to clarify sampling concepts and to provide a common ground for current and forthcoming research. In the classification criterion, a set of features related to sampling granularity, selection scheme and selection trigger are identified and proposed as the main components distinguishing current proposals. Then, each component is further divided into a set of approaches commonly followed in both classic and recently proposed sampling techniques. The proposed taxonomy allows a modular composition of sampling techniques, sustaining the definition of flexible and encompassing measurement strategies driven by packet sampling. The next chapter incorporates this classification scheme and its underlying modules into the proposal of a measurement architecture based on traffic sampling.

Chapter 5

Sampling-Based Measurement Architecture

As presented previously, the main objective of this work is to devise an encompassing, flexible and lightweight traffic sampling architecture aiming at supporting efficient sampling strategies adequate to diverse traffic scenarios and measurement activities. Taking the proposed taxonomy as a key enabler for the modular design of multiple sampling techniques, and thus complying with the specificities of each measurement scenario, this chapter proposes a three-layer measurement architecture in order to address the objective defined above. Each component of the architecture is described considering the different strategies and technologies that compose the several stages of a measurement process. A modular sampling framework prototype covering the main architectural components proposed is further presented in Chapter 6.

5.1 Design goals

Aiming at accomplishing the general objectives pointed out in Chapter 1, the definition of an encompassing, flexible and lightweight architecture for network measurements based on sampling addresses the following design goals:

- compatibility with current protocols and measurement tools in order to support its deployment in current measurement systems;
- specification and deployment sustained by open and standard protocols;
- flexibility to support the introduction of new protocols and technologies related to traffic sampling;

- versatility and modularity to deploy current and emerging sampling techniques;
- capability to support mechanisms for balancing measurement accuracy and computational weight in order to foster the design of efficient sampling strategies.

5.2 Architecture overview

The main components involved in the proposed sampling-based measurement architecture are arranged in three planes, as illustrated in Figure 5.1 and presented below:

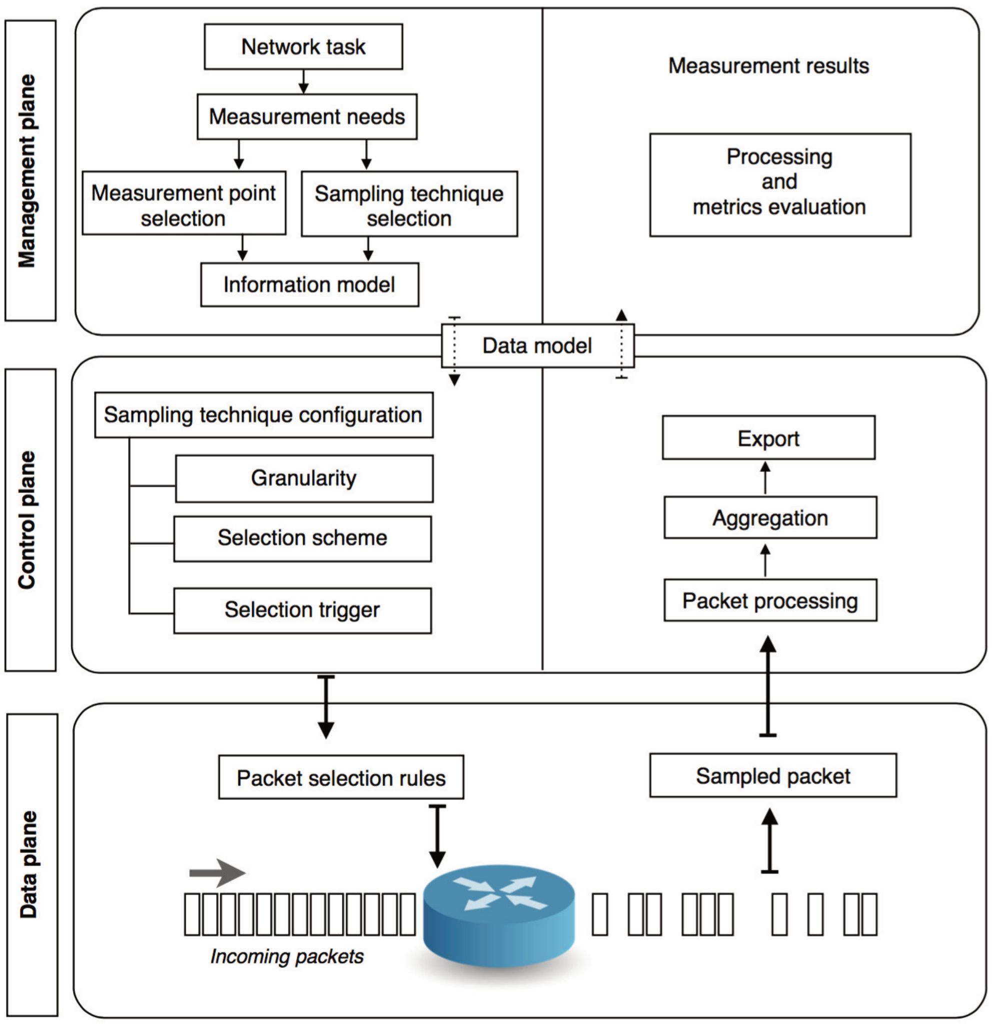


Figure 5.1: *Architecture description*

- The *management plane* includes tasks deployed directly in MPs or in external management entities. Based on requirements of each network task, measurement needs are identified, the more suitable sampling technique is designated and one or more MPs are selected to participate in the sampling process. This also involves identifying an information model able to define managed objects in the network independently of specific implementations or protocols in use.

Apart from providing the corresponding configuration parameters to the control plane, the management plane is also responsible for estimating the relevant metrics using data reports sent by the control plane. The required processing may involve results from single or multipoint measurements.

- A modular design of the *control plane* allows a flexible selection and configuration of sampling techniques. Considering the taxonomy presented in Chapter 4, the different approaches from each component can be arranged to compose the sampling technique designated by the management plane as the more suitable for a specific measurement scenario.

  In the control plane, sampled packets received from the data plane are processed and the relevant field contents are extracted according to the network task and measurement needs. These values are then aggregated (both in time and space) and exported following IETF IPFIX specifications (*i.e.*, RFC6728 [75]).

- At *data plane*, traffic is collected from network interfaces (*e.g.*, line cards or interfaces of packet forwarding devices) by applying the sample rules defined in the control plane. The unprocessed packets are then reported to the control plane to be processed, simplifying the data plane.

5.3 Management plane

The main activities assigned to the management plane are: (*i*) map the measurement needs related to a specific network task into the more suitable sampling technique and its operational parameters; (*ii*) select and communicate with the MPs which will perform packet sampling in order to set them up; (*iii*) process the measurement results and provide a visualization component, when applicable, based on reports produced by the control plane.

The functions that compose the management plane may be deployed directly into the MP, sharing the same device and resources, or in an external entity, responsible for coordinating one or more MPs according to measurement needs and constraints.

5.3.1 From network task to measurement needs

The measurement needs are closely related to the network task to fulfill. This relation usually guides the sampling process, defining aspects such as which portion of the packet should be captured and exported, if the sampling should be performed considering all incoming traffic or only specific flows, the temporal and spatial distribution of the packet collection or the expected accuracy on metrics estimation.

Some network tasks, such as traffic accounting, only require few information from the packet header, usually the key flow, the number of packets and number of bytes traversing the MP during a certain time interval. These requirements tend to be less impactful in terms of storage and bandwidth, as traffic may be aggregated into flows efficiently before exporting. However, considering network tasks which resort to Deep Packet Inspection (DPI), such as traffic classification and data analysis for security issues, the MP must inspect and collect the packet payload in addition to the header. This also involves exporting individual packets instead of aggregated summaries, which may lead to a large volume of data related to measurements transmission.

Even for the same network task, the measurement needs may vary depending on the expected accuracy in metric estimations, for instance, the accuracy in estimating the traffic workload is affected by the sampling frequency, where a higher number of captured packets lead to better estimation accuracy [71].

The relation between the network task and its measurement should drive directly the decision of the sampling technique and the MPs to be used. Despite this mapping being out of scope in this work, the next two sections present important aspects to be considered and highlights possible strategies.

5.3.2 Sampling technique selection

As discussed in Section 2.5, although there is not an encompassing study addressing which sampling technique yields better results for each network task, it is possible to identify that the results achieved by different works are clearly heterogeneous and, sometimes, conflicting. Associating the measurement needs with the most efficient sampling technique is particularly hard due to the heterogeneity of traffic scenarios and the lack of a platform able to fairly compare all current techniques.

Nevertheless, the related literature (covered in Section 2.5) may be used as initial input toward the decision on which technique shall be used in a specific scenario. Moreover, Chapter 7 will provide new insights on the performance of different techniques, which should be considered in this decision taking into account hardware features and constraints of the MP.

Considering that the compatibility with current measurement tools is an important design goal of this architecture, the choice of a sampling technique also must consider the sampling availability of those tools. Other aspect to be considered is the possible conflicting measurement needs from different network tasks performing sampling in the same MP during a time interval. This aspect may be addressed resorting to a priority system, in which the decision from the task with highest priority level prevails in the MP, or configuring the most demanding technique (*i.e.*, which captures the largest amount of data) in order to accommodate the largest possible number of network tasks efficiently.

Currently, due to the small number of sampling techniques available in measurement tools, the selection of the technique and its operational parameters is usually a decision of network managers. However, increasing the number of sampling techniques and attributes used to compare them (*e.g.*, computational weight) requires the development of new automatic systems based on strategies adjusted to specific network scenarios. In this way, a work currently in progress addresses strategies for automatic selection of sampling techniques resorting to ontologies, in which the entities *sampling technique* and *network task* are being semantically balanced regarding attributes such as measurement needs, estimation accuracy, overhead and performance.

5.3.3 Measurement point selection

The network task and its underlying measurement shall also drive the selection of the MPs that will participate in the sampling process, which may involve a single point, a dual point (*e.g.*, end-to-end delay) or a distributed multipoint strategy. Considering the design goal that the measurement architecture should be compatible with existing measurement systems, the MP selected may also be a sampling tool embedded in network devices (*e.g.*, routers, switches and firewalls), NICs or a stand-alone device set to perform packet sampling. Details about collecting packets from the network are presented in Section 5.5.

In this way, the decision on which MPs will participate in the measurements takes into account the position of the device in the network, the computational resources available and the sampling techniques able to be deployed (for legacy tools). Figure 5.2 illustrates a network domain with five MPs available. As examples of the relation between the network task and the MPs selection: (*i*) traffic accounting in this topology could involve only the MP-A (sampling the external link of the border router); (*ii*) QoS multipoint metrics estimation, such as one-way-delay and jitter, would involve the edge MPs, for instance MP-C and MP-E; (*iii*) security-oriented measurements would probably require data sampled by all MPs.

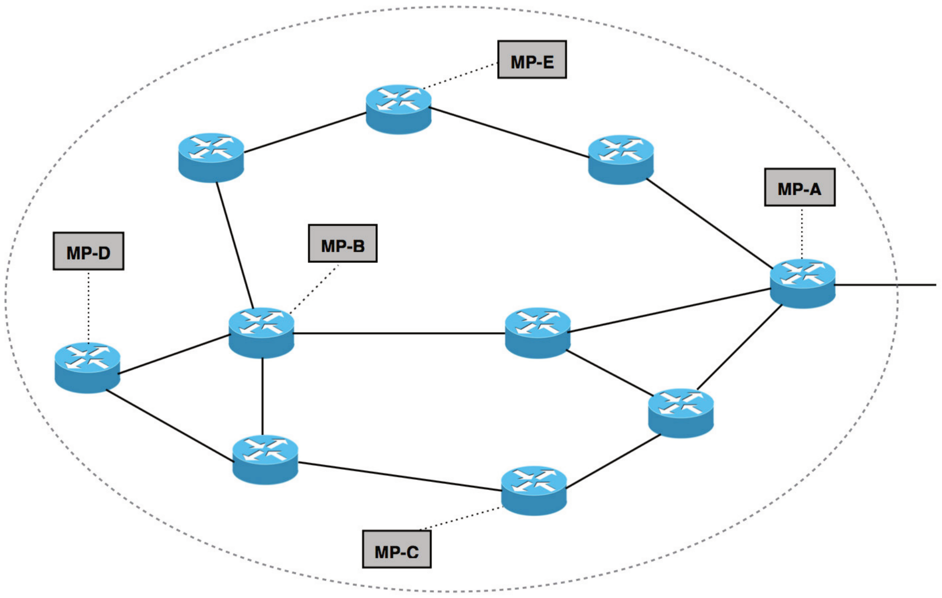


Figure 5.2: *Measurement point selection*

There are many studies addressing the selection of the better network point to perform each type of measurements [76] [77]. An appropriate strategy of selecting the most suitable MP regarding the network task leads to more efficiency in the resources usage and may reduce the events of conflicting configuration.

5.3.4 Information model

As described in RFC3444 [78], the main purpose of an information model is to define managed objects at a conceptual level, independently of specific implementation or protocol used to transport data. The level of abstractions depends on the modeling needs, however, it should hide all protocol and implementation details in order to make the overall design as clear as possible. In this way, the information model is defined at management plane as a standardized way for encoding information related to the sampling process (*e.g.*, technique selected, sampling parameters and packet fields to be collected), exporting and storage of the sampled data.

As result of the efforts toward the definition of an open protocol for flow exporting [64], the IETF IPFIX working group has also produced a standard defining an information model (*i.e.*, RFC7012 [79]) that was further extended to satisfy PSAMP requirements through RFC5477 [80]. Proposing an extended model was necessary due to existing properties required in packet sampling reports that cannot be modeled using the basic IPFIX information model.

The information model is composed by unique identifiers related to each *information element*, that consists in an encoding-independent description of an attribute that may appear in a measurement record. The information elements also have an associated type, that indicates constraints on what it may contain as well as the valid encoding mechanisms [79]. The information element assignments are controlled by IANA - *Internet Assigned Numbers Authority* which provides flexibility to introduce additional elements, such as new techniques deployed following the taxonomy presented in Chapter 4.

Table 5.1 presents some elements defined by IANA. As an example, the element *selectorAlgorithm*, identified as *304*, corresponds to the sampling technique in use and must be encoded as an *unsigned32* (*i.e.*, non-negative integer with 32 bits). A full list of elements currently assigned can be consulted in [81].

Table 5.1: *Example of IPFIX information elements.*

Element ID	**Name**	**Data type**	**Data type semantics**
1	octetDeltaCount	unsigned64	deltaCounter
2	packetDeltaCount	unsigned64	deltaCounter
304	selectorAlgorithm	unsigned16	identifier
309	samplingSize	unsigned32	quantity

5.3.5 Data model

Data models define managed objects at a lower level of abstraction, including implementation aspects and protocol specifications, such as the rules that explain how to map managed objects onto lower-level protocol constructs [78]. The definition of a common data model is required to allow managing the entities in the sampling process.

In addition to the specific information model for packet sampling, the IETF IPFIX working group has also proposed a standard that defines managed objects for monitoring devices performing packet selection by sampling (*i.e.*, RFC6727 [82]). The document is in accordance with the Internet-Standard Management Framework [83], in which managed objects are stored in a MIB and generally accessed through SNMP. The syntax used to define objects in the MIB is called the *Structure of Management Information* (SMI). Currently, the working group is defining a standard with a method for exporting SNMP MIB variables using IPFIX messages [84].

There are other management models currently standardized as data models, for instance the *Policy Information Base* (PIB) [85] and the *Common Information Model* (CIM) Schemas [86]. However, the use of open standard models designed to support packet sampling leads to a straightforward integration with current tools able to control MPs and process the resulting sampled data reports.

To foster the compatibility with a large number of measurement applications, the deployment of the proposed architecture is expected to include the capability to export reports in XML - (*Extensible Markup Language*), due to its comprehensive support, simplicity, generality and usability.

5.3.6 Processing and metrics evaluation

As discussed in Section 5.4, the packet sampling process yields different types of reports that must be processed for further estimation of underlying metrics regarding the network task. This is usually performed by a *collector*[1], that may be an intermediate entity responsible for verifying and distributing the reports to interested entities or an application able to provide summarized measurement results.

Following IETF definitions [1], the collector receives a *report stream* by one or more MPs. The report stream comprises two types of information: (*i*) *packet reports* - a configurable subset of packet's data regarding the measurement needs (*e.g.*, packet content); and (*ii*) *report interpretation* - a subsidiary information used for interpretation of the packet reports (*e.g.*, templates describing its structure and types). An example of both report types is presented in Section 5.4.4 (Figure 5.4).

Supported by the definition of a consistent information model and data model, the entities involved in sampling can process the reports and access the required traffic information in order to estimate measurement metrics. This mechanism also allows the compatibility with several monitoring applications designed for specific network tasks.

Strategies for metrics' estimation toward a particular network task are beyond the scope of this work, however, some methods related to traffic accounting, characterization and classification are presented and discussed along Chapter 7.

[1]In this work, a *collector* consists in an entity or process capable of receiving an processing IPFIX messages.

5.4 Control plane

The control plane consists in the main component of the proposed architecture. It is responsible for: (*i*) selecting and arranging the taxonomy approaches (see Chapter 4) in order to deploy the sampling technique defined by the management plane and setting its operational parameters; (*ii*) receiving raw packets collected by the data plane and extract required information regarding the measurement needs; (*iii*) aggregating data in order to reduce the storage and transmission impact; (*iv*) composing the appropriate reports to be sent to the management plane.

5.4.1 Sampling technique configuration

Following the guidelines presented in Section 5.3.4, the MP receives the necessary information in order to select and configure the sampling technique that will supply the management plane according to the network task requirements. Table 5.2 shows examples of sampling techniques, their underlying operational parameters and the respective information element identification according to the IANA scope assignments.

Table 5.2: *Sampling technique identification example.*

ID	Sampling technique (selectorAlgorithm (304))	Parameters (ID)
1	Systematic count-based	samplingPacketInterval (305) samplingPacketSpace (306)
2	Systematic time-based	samplingTimeInterval (307) samplingTimeSpace (308)
3	Random n-out-of-N	sampleSize (309) samplePopulation (310)
10[1]	Multiadaptive	samplingTimeInterval (307) samplingTimeSpace (308)
11[1]	Flow-level adaptive linear prediction	flowId (148) samplingTimeInterval (307) samplingTimeSpace (308)

As presented in Table 5.2, the sampling technique selected is identified by the information element *selectorAlgorithm*, identified in the information model with the value *304* (detailed in Section 5.3.4). Each sampling technique has a set of well-know parameters (also defined in the information model), that must be passed along with the technique identifier. Handling this information, the control plane starts an instance of the technique with the respective operational parameters.

[1]As these techniques are not yet assigned by IANA, the examples use currently unassigned values, avoiding conflicts with deployed tools.

The internal process for deploying a sampling technique follows the structure of the taxonomy proposed in Chapter 4. By selecting the appropriate approach from each sampling component, the control plane arranges the respective blocks, configuring thus the selected technique. This process requires well defined communication interfaces among the sampling approaches, that are achieved designing the *sampling framework* as a multilayer system in which a lower layer provides services to an upper layer, hiding details about its operation. Packet capturing is performed resorting to an interface with the data plane and its operational details are covered in Section 5.5. Figure 5.3 illustrates the conceptual design of the framework, while Section 6.2 presents the details regarding its implementation.

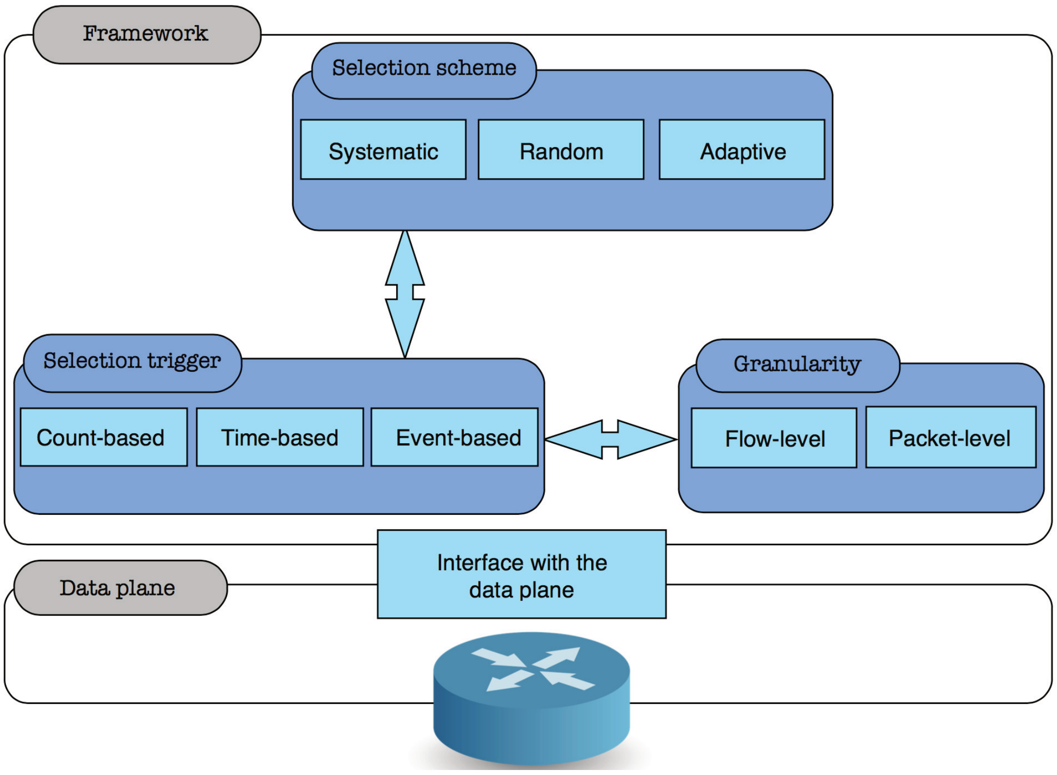


Figure 5.3: *Sampling framework - conceptual design*

A work in progress related to this doctoral work, intends to extend the flexibility introduced by the modular composition of sampling techniques through the recent developments in SDN (Software-Defined Networking) research field. The principle behind SDN architecture is that the control and data plane of the network nodes are decoupled, with a centralized logic controller and view, abstracting the underlying network infrastructure to applications. To support it, open interfaces between the devices in the control plane and those in the data plane provide programmability of the network behavior by external applications [87]. Although firstly oriented to packet switching, network measurements have also emerged as one promising field for SDN [88]. In this way, the main idea is to exploit SDN flexibility in order to enable programmable measurements and thus, allowing to introduce the sampling taxonomy principles into SDN controllers.

5.4.2 Packet processing

As discussed in Section 5.5, in order to reduce the computational burden in the data plane, the selected packets are received by the control plane in raw format, for verification and processing.

The verification process allows to identify errors in the packet that may have occurred during the handover from the capture interface to the upper plane of the network stack. Error detection can be performed resorting to any available method, such as checksum, parity bits or cyclic redundancy checking. As error correction is a computationally onerous process, if an error is found, than the packet is discarded.

Considering that packets are received in raw format, it is also necessary to map the packet fields to the suitable data model in use by the measurement system. The mapping process is supported by the information model in order to unify the elements representation, allowing the correct interpretation and manipulation of the packet fields.

At the processing stage, all irrelevant fields to measurements are discarded, reducing the amount of data received and processed by the upper modules, and consequently the number of computation cycles, bandwidth and memory to process the sampled traffic.

5.4.3 Aggregation

During the sampling process, a subset of all packets traversing a monitored link is selected and further transmitted to a collector or an application in order to estimate relevant metrics for the network task. However, even with the reduction in the volume of data promoted by sampling, some scenarios can still produce massive data amounts, requiring significant storage resources and bandwidth to be distributed.

A solution to this issue is to summarize the collected data employing a combination of sampling and aggregation. Thus, as mentioned before, aggregation leads to a loss in data resolution for analysis. In this way, the usage and the level of aggregation must also be aligned with measurement needs.

The strategy mostly used to summarize measurement data is to aggregate sampled packets into flows according to some explicit or derived property, and computing aggregate byte and/or packet counts within each flow over successive time windows [89]. This method usually provides high accuracy when estimating metrics that only require information from the packet header, such as for traffic accounting, flows distribution and anomaly detection.

However, there is an increasing need for analyzing packet payloads (*i.e.*, DPI) in network tasks such as traffic classification and security analysis. In fact, the IANA has been registering information elements describing objects related to the application layer. In these scenarios, as packets are typically stored as individual entities, and thus hampering data summarization, the aggregation module does not act on packets, maintaining them in memory for future exporting.

Additionally to the measurement requirements, the differences regarding the performance of transmission and storage resources must be taken into account (see Section 5.4.4). Some of these constraints can be handled resorting to an effective management of the period in which the aggregation process stores the data before dispatching, which evince the close relation between the aggregation module and the exporting module.

5.4.4 Exporting

After the selection and capture, the sampled packets are aggregated (when feasible) and stored for exporting to a collector or an application that will estimate the required parameters. The exporting process consists of three main elements: (*i*) the trigger for dispatching data currently stored in the MP memory; (*ii*) the message structure and types used to transmit the data; (*iii*) the transport protocol used in the data report transmission.

Exporting trigger

Along the sampling process, the MP must satisfy one or more conditions defining when the measured data stored in its memory should be transmitted. These conditions are usually related with the aggregation strategy and measurement requirements.

Sampled traffic aggregated into flows is frequently maintained in memory until a specific flow is considered to have terminated, after which, the information regarding this specific flow is exported. The flow expiration may follow different methods: a natural termination of a TCP flow when a packet with a FIN or RST flag set is captured; the flow has been active for a specific period of time (usually within the range between 120 seconds to 30 minutes); or no packet belonging to a flow is captured during a specified period of time (usually between 15 seconds and 5 minutes) [90]. In addition to predefined timeouts, resource constraints may require strategies in which the timeout is dynamically adjusted in run-time.

Considering measurements involving the exporting of the packet payload, the methods explained above may not be sufficient, requiring alternative options in order to optimize the exporting process. In this case, the MP may also use predefined timeouts or a dynamic strategy based on the volume of packets stored in memory. Moreover, this strategy is also suitable to trigger the exporting of both aggregate and single packet entries in scenarios of full memory or in response to unexpected situations.

Message format

As discussed in Sections 5.3.4 and 5.3.5 respectively, the information model provides the conceptual models to identify each element involved in a sampling process, while the data model defines how to map them onto lower-level protocols. Furthermore, beyond these definitions, the distribution of sampled data must be supported by well defined message formats in order to ensure that the applications involved in measurements can interpret correctly the information.

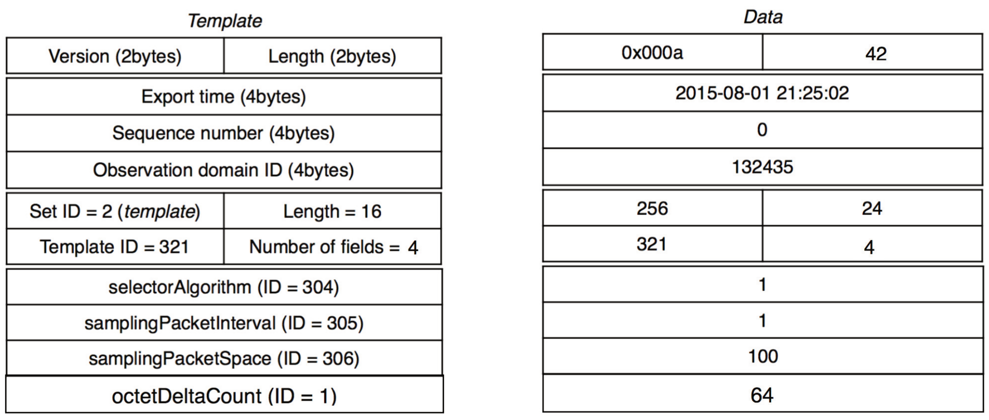
In this way, there are well defined protocols and tools able to handle with this aspect, such as NetFlow, SiLK and IPFIX. Although these are not the only possible solutions (reports using XML specifications are also frequently used), the IPFIX protocol is the most adequate option to support the flexible way to deploy sampling techniques proposed in this work.

In fact, there is a specific version of IPFIX specification (*i.e.*, RFC5476 [91]) designed to address the architectural differences between the original version (*i.e.*, RFC7011 [64]), focused on gathering and exporting IP traffic flow information, and the PSAMP extension, focused on exporting information of individual packets. Sampling-based reports are therefore a special IPFIX record containing only a single packet.

Basically, the IPFIX message can be of two types: (*i*) *template record*, that contains the layout description for data report interpretation; and (*ii*) *data report*, used for carrying exported data records [90]. For each collected packet, a data report must be created containing a header with a set of fields with fixed size (16 bytes), identifying the protocol version number, message length, export timestamp and the observation domain ID. After the header, one or more sets (*i.e.*, one or more records), are defined having an ID and variable fields.

Figure 5.4 presents a simplified example of a template record message and its correspondent data report, identifying that the packet, with 64 bytes, was collected through *systematic count-based* sampling technique capturing 1 packet from every 100. All elements in a report are specified by the information model (see Section 5.3.4) and each template has an unique ID, allowing that all entities involved in the measurements can interpret the data reports following this template.

Usually the number of records in an IPFIX message is limited aiming at avoiding IP fragmentation. In this way, the exporting module decides how many records will be included in the message ensuring that the message size does not exceed the Maximum Transmission Unit (MTU), excepting the cases in which information elements with variable lengths exceed the link MTU [90].



Template

Version (2bytes)	Length (2bytes)
Export time (4bytes)	
Sequence number (4bytes)	
Observation domain ID (4bytes)	
Set ID = 2 (*template*)	Length = 16
Template ID = 321	Number of fields = 4
selectorAlgorithm (ID = 304)	
samplingPacketInterval (ID = 305)	
samplingPacketSpace (ID = 306)	
octetDeltaCount (ID = 1)	

Data

0x000a	42
2015-08-01 21:25:02	
0	
132435	
256	24
321	4
1	
1	
100	
64	

(a) Template record (b) Data report

Figure 5.4: *IPFIX messages - template and data.*

Transport protocol

In addition to the message format, the exporting process must also select a transport protocol to transmit the measurement reports. This decision may be taken regarding the collector/application or MP restrictions. The usual candidates are UDP - User Datagram Protocol, TCP - Transmission Control Protocol and SCTP - Stream Control Transmission Protocol.

Due to the easy implementation (even in hardware) and minimal overhead, UDP is the most implemented transport protocol for measurement data transmission. However, possible drawbacks are: (*i*) the lack of congestion control may incur in significant loss of messages, mainly in high-volume exports; (*ii*) as the protocol does not provide flow control, the collectors and applications must use large socket buffers in order to handle bursty reports; (*iii*) when exporting based on report templates, the exporting module must periodically resend templates to ensure that all involved entities have received them and thus interpret correctly the data reports, which require mechanisms to handle lost and duplicate messages [90].

Considering a sampling based scenario, in which losing one report may imply the loss of all information regarding a flow, the congestion-awareness and reliability provided by TCP turn it a better choice when compared to UDP. The primary problem with TCP is that the receive window mechanism may limit the exporting process, forcing it to maintain the messages in the transmission buffer, which increases the risk of discarding packets that exceed the sender capacity.

This issue can be addressed resorting to SCTP that provides multiple streams per collection allowing to separate logically the exported information simultaneously [92]. Furthermore, the sender capacity to cancel retransmission of unreceived segments after a given timeout allows selective dropping of exported segments under high load, rather than overloading buffers with pending retransmissions [90]. However, SCTP is currently the least deployed of the three protocols mainly due to its difficulty to be implemented.

5.5 Data plane

At data plane, following the sampling rules defined in the control plane, packets are collected from the network link for subsequent use. This section includes a description of the technologies and mechanisms to perform packet capture, that may be deployed in wired, wireless or virtual networks. Due to performance issues involved in reading packets from network links, mainly in high-capacity networks, the processes implemented in this plane must be kept simple, avoiding processing overhead.

In wired networks, where most traffic measurements are performed, the MP implements an interface (also called capture device) in which it is possible reading and collecting packets from the link being monitored. The capture interface can be positioned *in-line* and in *mirroring* mode. While in in-line mode, the MP is directly connected to the monitored link between two hosts, usually resorting to a network tap that duplicates all observed traffic through passive splitting (on optical fiber links) or regeneration (in electrical copper networks), in mirroring mode, the network device forwarding packets can mirror packets from one or more ports to another port, in which the MP device is attached.

In wireless networks, the MP may use any device with a compatible interface (usually these devices can only capture packets at a single frequency at a given time [90]), however some of them can switch rapidly through all radio channels (channel hopping) trying to improve traffic capturing, although there is no guarantee that all packets are considered [93]. In virtual networks, the nature of the devices is similar to wired networks, although in this case the capture interfaces are usually entirely deployed in software.

Basically, before any packet pre-processing, packets must be read from the capture interface, that may be a NIC or any other programable device implementing a compatible network stack. After selecting which packets will compose the sampling, they are collected and timestamped. Then, the packets are reported to the control plane to be processed.

In addition to the device nature and location, the data plane also defines how the control plane interacts with the network interface in which the packet capture is performed. For this, MPs resort to libraries and Application Programming Interfaces (APIs) in order to implement the packet collection. The main solutions currently available are using *libpcap* [94] or *libtrace* [95] for Linux and BSD-based operating systems, and *WinPack* for Windows.

Considering that packets have to traverse several layers from the interface to the library (which is located at the top of the operating system's network stack), the overall capture performance depends on the efficiency in handing over packets from the capture device interface to the upper plane via the packet capture library. One of the various strategies proposed to improve this process is using a memory mapping technique in order to reduce the cost of copying packets from kernel-space to user-space through Direct Memory Access (DMA) [90].

5.6 Distributed measurement systems

This section provides insights regarding strategies to deploy a distributed measurement system based on the architecture presented above.

For networks with several MPs, configuring sampling techniques and exporting measured data may incur in some potential problems. Figure 5.5 presents a distributed environment in which some of these problems can be evinced. In this illustration, three different measurement tasks running in distinct points of the network require measurement information from various MPs distributed through the network.

As discussed earlier in this chapter, each measurement task poses possibly distinct measurement needs, which can be satisfied using the more suitable sampling technique available in the modular sampling framework (detailed in Chapter 6). Although conflicting measurement needs can be addressed resorting to a proper priority system or configuring the most comprehensive technique (see Section 5.3.2), the tasks toward this decision are still performed by the MP. This can be heavy in terms of computational resource consumption, particularly in low power devices.

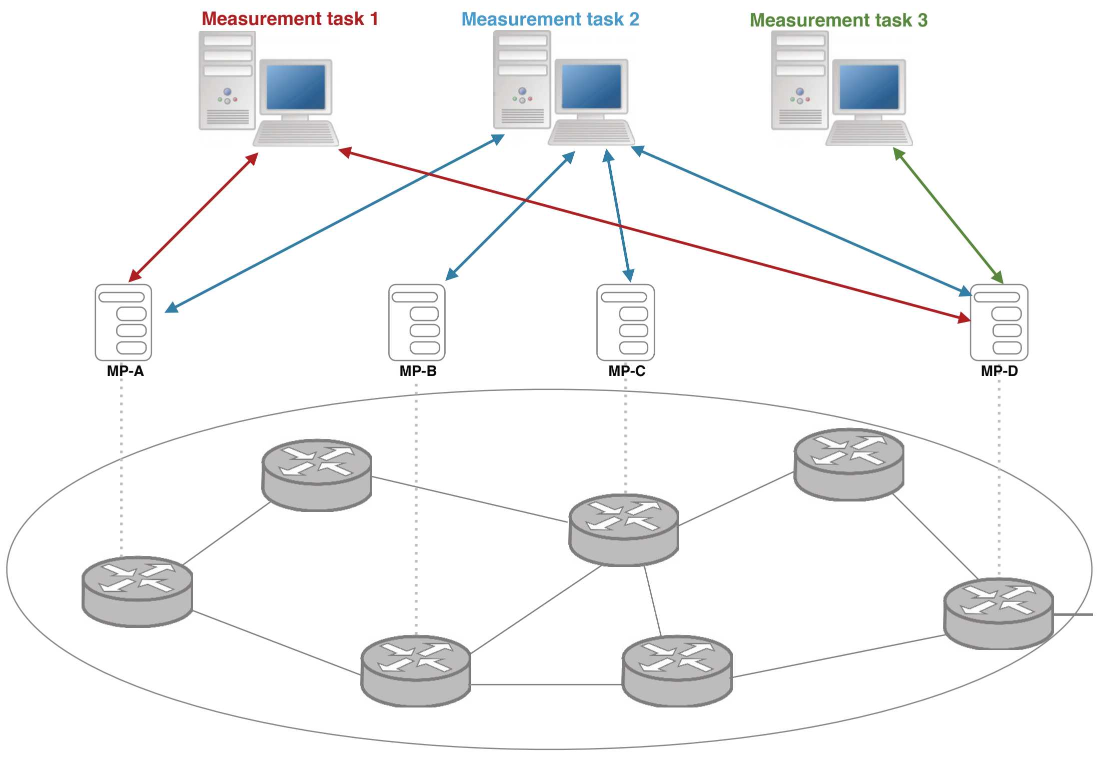


Figure 5.5: *Measurement exporting - unicast*

Other potential drawback in the environment presented in Figure 5.5 is related to exporting several copies of the same measurement report. Having defined which sampling technique and parameters will be performed, each MP needs to export one copy of each report to all applications or collectors involved in the measurement. Those copies may overflow the MP buffer, leading to lose report messages, further impacting on the overall performance of devices sharing exporting and capturing interfaces.

In order to handling with these potential problems, this work advocates the usage of a central entity able to intermediate MP configuration and multicast dissemination for attenuating the impact of message copies in the MP.

A distributed measurement system centrally controlled can be achieved introducing an external entity able to mediate the MPs configuration and exporting destination. This *Central manager* is therefore responsible for: (*i*) authenticating the applications interested in measurement data; (*ii*) defining the priority of each measurement task;

(*iii*) selecting and configuring the best sampling technique in each MP regarding the measurement needs; and (*iv*) coordinating the report distribution in order to save resources. Although a *Central manager* is in place, the respective services may still be deployed in a distributed way, aiming at addressing scalability, security and reliability issues.

Figure 5.6 illustrates a distributed measurement system based on the sampling framework in presence of the central manager. Its general operation is presented below:

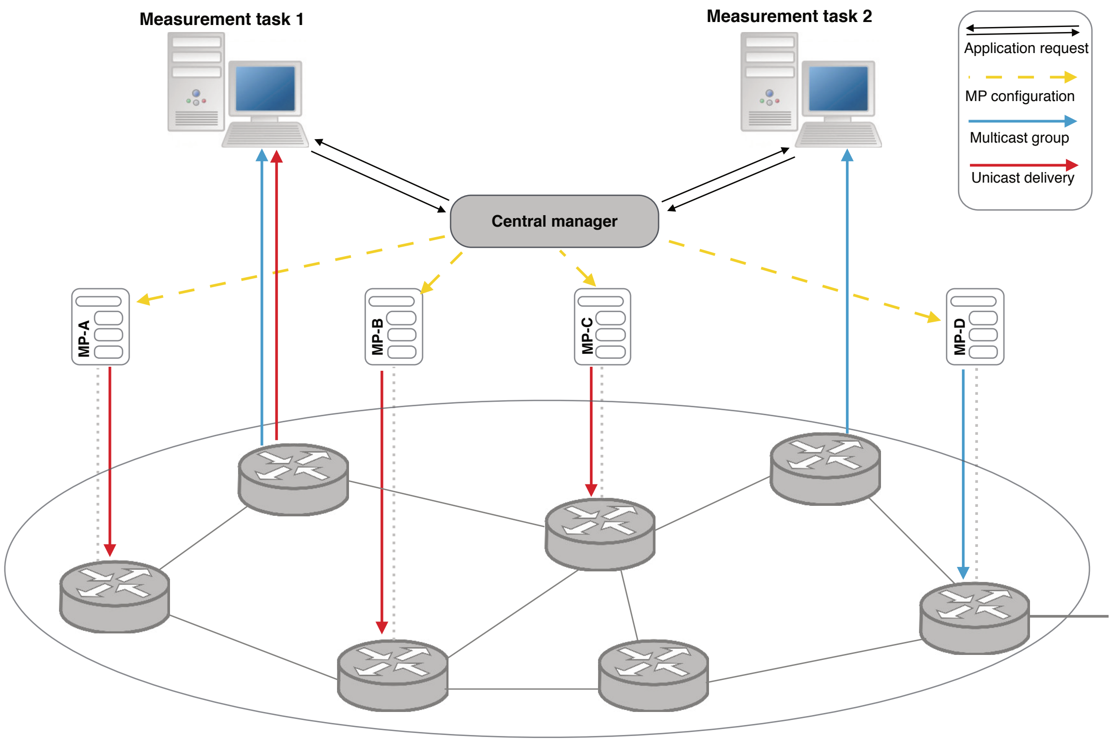


Figure 5.6: *Measurement exporting - multicast*

- when an application performing a measurement task is interested in some measurement results, it sends a request to the central manager specifying the identification of the specific MP or group of MPs of interest, the most suitable sampling technique and its underlying configuration parameters;
- if the selected MPs are not involved in any measurement process, the central manager configures them with the indicated parameters and the unicast address to which reports should be sent to. This scenario is represented in Figure 5.6 by red arrows, in which the MPs A, B and C are exporting their results to a single destination (*i.e.*, Measurement task 1);

- on the other hand, if the selected MPs are already supplying other measurement tasks, the central manager firstly analyzes which task has the highest priority in defining the measurement configuration parameters. Then, it sets all MPs with the most priority parameters or with the more comprehensive approach, aiming at satisfying the requirements from measurement tasks with equal priority;
- in addition, the central manager configures a multicast group and notifies all MPs and applications. The applications use the multicast address to join the group to which the measurement results will be sent, and the MPs use this address as a destination of their results, without joining the group. This approach avoids that one MP receives reports from other MP. This scenario is also represented in Figure 5.6, in which the blue arrows correspond to a multicast group with the MP D exporting its results to the Measurement tasks 1 and 2.

In this proposed approach, the central manager only coordinates the set up of the entities (*i.e.*, MPs and applications performing measurement tasks) involved in the measurement process. After that the datagrams containing measurement reports are directly delivered to the applications, without intervention from the central manager until other application expresses interest in receiving the measurement results from a MP under its control.

Resorting to multicast communications aim at attenuating the impact of report copies on MP interfaces and reducing the total bandwidth consumed during the exporting process. For networks in which IP multicast is not supported, alternatives such as overlay multicast (multicast-over-unicast) or Explicit Multi-Unicast (Xcast) [96] can be addressed to provide efficient group communications.

5.7 Summary

This chapter has presented an encompassing and modular measurement architecture able to support the design and deployment of flexible sampling-based measurement systems. The main components involved in the proposed sampling-based architecture are arranged in three planes. The *management plane* includes tasks related to mapping measurement needs into the more suitable sampling technique to be deployed in one or more MP. Apart from providing the corresponding configuration parameters, the management plane is also responsible for receiving sampled data, used to estimate the relevant metrics according to the measurement task. The *control plane*, according to the proposed taxonomy, supports the modular definition of classic and new sampling techniques and configurations, that can be adjusted to each traffic/service measurement scenario. At this plane, the sampled packets are processed and the relevant fields content are extracted according to the network task measurement needs. These values are then aggregated (both in time and space) and exported. At *data plane*, traffic is collected from network interfaces by applying the sample rules defined in the control plane. The unprocessed packets are then reported to the control plane to be processed, simplifying the data plane. In addition, a strategy for the deployment of a distributed measurement system has been presented, including the entities involved. The implementation details of a prototype based on the proposed architecture are presented in the next chapter.

Chapter 6

Sampling Framework Prototype

Attending to the main goal of providing an encompassing measurement system able to foster the design and deployment of flexible sampling strategies, this chapter describes the implementation of a sampling framework prototype including the main architectural components described in Chapter 5 and their functional interactions.

The developed framework is then used as a proof-of-concept regarding the flexibility introduced by the sampling taxonomy and as a fair environment to perform comparative assessments involving different sampling techniques in diverse traffic scenarios. This analysis aims at fostering the introduction of new sampling strategies adjusted to specific measurement needs, optimizing thus traffic measurements.

6.1 Technological aspects

As discussed in Section 2.5, most of the current measurement tools able to perform packet sampling only deliver a limited set of sampling techniques. They also do not provide mechanisms able to support the integration of new proposals. Therefore, the development of the sampling-based architecture's core (*i.e.*, the sampling framework), must be sustained by open and well accepted technologies able to be easily deployed in different MP architectures, reducing thus the dependency of proprietary solutions. This design directive allows not only selecting the best sampling technique regarding specific traffic scenarios and measurement needs, but it also enables the possibility to address the most suitable hardware, avoiding the use of expensive and dedicated devices.

In this way, regarding the packet capture at MPs interfaces, the technology adopted is *libpcap* [94], an open source and portable packet capture library widely used by the data networking community and available for most operating systems[1]. It is also the core of several packet capture tools and traffic analyzers, such as *tcpdump* [97], *dsniff* [98], *snort* [99] and *ettercap* [100], which assures a large support and research addressing performance issues. In this regard, due to the simplicity of the data plane, which is responsible only for packet capture and timestamping, the overall performance depends upon MPs hardware and their operating system [90]. Furthermore, *libpcap* output format (*i.e., pcap*) is widely supported by exporting systems, such as *yaf* [101] which is compatible with the IPFIX standard.

Other advantage of using *libpcap* is that, although it was developed to be used along with C and C++, there are various APIs that allow its use from several programming languages, for instance, Perl, Python, Java, C# and Ruby. In this way, considering the portability goal, the language elected for the framework prototyping is Java due to its extensive support in different computer architectures, running in all of them from a unique code compilation. In this study, the specific API used to control *libpcap* from Java is JPCAP [102]. Despite the Java Virtual Machine (JVM) being usually associated with low performance, there has been specific research in which the bytecode processing is being improved toward efficient integration with *libpcap* [103].

Sustained by the flexibility and comprehensiveness of the technologies above mentioned, the developed sampling framework is currently deployed as an experimental monitoring system. It is being applied into various network scenarios and measurement tasks, running on different computer architectures, namely x86, x64 and ARM (detailed in Section 6.3).

6.2 Framework structure description

The sampling framework is developed following the taxonomy presented in Chapter 4 and the relationship among its components. The framework design can be seen as a multilayer system in which a lower layer provides services to an upper layer, hiding details about its operation. The resulting framework allows to combine the sampling components defined in the taxonomy, providing a flexible platform that can be applied to online and offline measurement scenarios. In addition, the technologies supporting its development and the preference for using standard protocols allow integrating the framework in different measurement systems easily.

Figure 6.1 presents an integrated diagram of classes from which the implementation

[1]Windows uses a port of libpcap known as *WinPcap*.

details of each taxonomy component and their underlying approaches are explained, including the communication mechanisms.

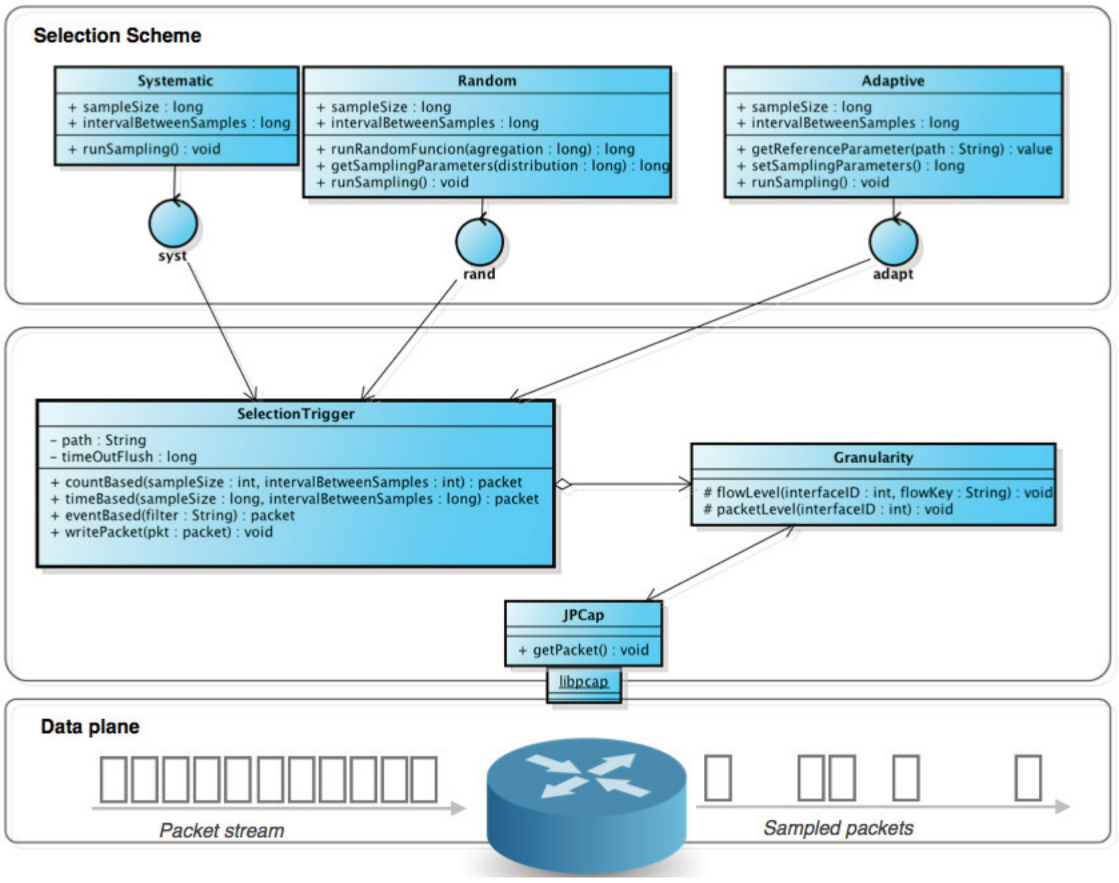

Selection Scheme
Systematic
+ sampleSize : long
+ intervalBetweenSamples : long
+ runSampling() : void
Random
+ sampleSize : long
+ intervalBetweenSamples : long
+ runRandomFuncion(agregation : long) : long
+ getSamplingParameters(distribution : long) : long
+ runSampling() : void
Adaptive
+ sampleSize : long
+ intervalBetweenSamples : long
+ getReferenceParameter(path : String) : value
+ setSamplingParameters() : long
+ runSampling() : void
syst
rand
adapt
SelectionTrigger
- path : String
- timeOutFlush : long
+ countBased(sampleSize : int, intervalBetweenSamples : int) : packet
+ timeBased(sampleSize : long, intervalBetweenSamples : long) : packet
+ eventBased(filter : String) : packet
+ writePacket(pkt : packet) : void
Granularity
# flowLevel(interfaceID : int, flowKey : String) : void
# packetLevel(interfaceID : int) : void
JPCap
+ getPacket() : void
libpcap
Data plane
Packet stream
Sampled packets

Figure 6.1: *Main classes in the sampling framework*

6.2.1 Granularity component

As presented in Figure 6.1, the *granularity* component is deployed in a single class and each of its different approaches correspond to a single method. This class is the first mechanism above *libpcap* and works controlling which packets will be considered for the next sampling steps.

The *flowLevel* method receives two parameters, namely the *interfaceID* and the *flowKey*. The first parameter consists in an *integer* identifying the MP network interface in which packet sampling will be performed. The list of available interfaces can be consulted from the main class resorting to the method *getInterfaces()*. The parameter *flowKey* identifies the flow or flows from which the packets will be sampled. Aiming at providing flexibility in defining flows beyond the classic 5-tuple scheme, the parameter is defined as a *String* following the *tcpdump* filter syntax[2].

Considering that in the *packet-level* approach all incoming packets are eligible for sampling, the corresponding method, called *packetLevel*, only receives the interface identification as parameter *i.e., interfaceID*. Thus all packets are forwarded to the upper layer of the framework, *i.e.*, the *selection trigger*.

6.2.2 Selection trigger component

Similarly, the *selection trigger* component is also deployed in a single class (*i.e., SelectionTrigger*) with individual methods implementing each approach. Regarding its operation, each method invocation starts a single sample collection. This allows to keep the class unchanged when applied in different sampling techniques, such as in adaptive techniques, in which the sampling frequency might vary during the measurement process.

As presented in Figure 6.1, the *countBased* method receives two parameters, *i.e.*, the *intervalBetweenSamples* and the *sampleSize*. These parameters correspond, respectively, to the number of packets ignored for measurement purposes, therefore not collected, and the number of packets collected to compose a sample.

The *timeBased* method receives similar parameters (*i.e.*, *intervalBetweenSamples* and *sampleSize*), however in this case they correspond to the timestamps (in milliseconds[3]) of the packets arrival at the MP. In this way, they are defined as *long integers*.

The *eventBased* method receives the parameter *filter* defined as a string that indicates which packet fields must be matched in order to capture specific ranges of packets. This parameter also follows the *tcpdump* filter syntax.

All the three methods implementing the selection trigger approaches return objects of type *packet*. This object type corresponds to a single instance of a packet following the *pcap* format. The selected packets are stored in a default location (*i.e.*, /pkt_temp/), resorting to the method *writePacket()*, for future aggregation and exporting. The class also provides a constructor to define an alternative destination for the sampled packets (*i.e., path*) and the time interval (*i.e., timeOutFlush*) between consecutive flushes. This corresponds to a mechanism to avoid an external process trying to read the temporary file while the framework is writing new sampled packets. In this way, different temporary

[2]A comprehensive coverage of the filter syntax can be found in http://www.tcpdump.org/manpages/pcap-filter.7.html

[3]Depending on the computer architecture in which the MP is deployed, the timestamp precision can be increased to microseconds.

pcap files are created every *timeOutFlush* period, defined by default as five minutes. The external processes can thus use this time to schedule the next operation cycle.

6.2.3 Selection scheme component

Considering that the *selection scheme* component corresponds to the main distinguishing feature among sampling techniques, involving possibly complex functions, each approach in this component is implemented as an individual class. This enhances the flexibility when deploying new techniques, as the methods within the selection trigger and granularity components are kept invariable, as discussed before.

The *systematic* approach, as defined in RFC 5475 [1], comprises the simplest sampling techniques and consists in successive invocations of the same method from the selection trigger object using invariable parameters. In this way, in the *Systematic* class, the method *runSampling()* selects the correspondent methods from the classes *SelectionTrigger* and *Granularity* in order to compose a specific systematic technique.

The *random* approach includes a random generator method which may follow different probabilistic functions, as explained in Section 2.2. Considering that the portion of the traffic collected varies in every sampling iteration, each invocation of the specific selection trigger method receives different parameters. Note that the sample size does not change, only its temporal or spatial position does. The method presented in the *Random* class in Figure 6.1 corresponds to the *random n-out-of-N* technique, discussed in Section 2.2.1, and is currently available in the sampling framework (see Section 6.3). Different random functions can be introduced by developing new classes with the respective mechanisms.

The *adaptive* approach is usually the most complex within the selection scheme component as it requires monitoring a reference parameter (*e.g.*, *throughput*) that will guide the sampling adaptiveness. As discussed in Section 2.4, adaptive sampling techniques also resort to a controller designed to analyze the reference parameter in order to make decisions on the sampling policy. This is accomplished through specific method invocations from the selection trigger object varying the sample distribution and/or the sample size. The *Adaptive* class presented in Figure 6.1 shows the corresponding methods that compose the *multiadaptive* technique, introduced in Chapter 3, in which both the sample size and the interval between samples can change dynamically along the sampling process.

6.3 Experimental framework prototype

The current version of the framework provides a wide number of sampling techniques, most of them unavailable in research or commercial tools. All the current sampling techniques deployed in the framework are listed in Table 6.1. They can be performed both in online and offline measurement scenarios. The offline mode consists in applying the packet sampling to traffic previously collected by any application and stored in *pcap* file format. In order to foster the introduction of new sampling strategies and to provide a platform in which these strategies can be fairly assessed, the framework source code is publicly available[4].

Table 6.1: *Sampling techniques available in the framework*

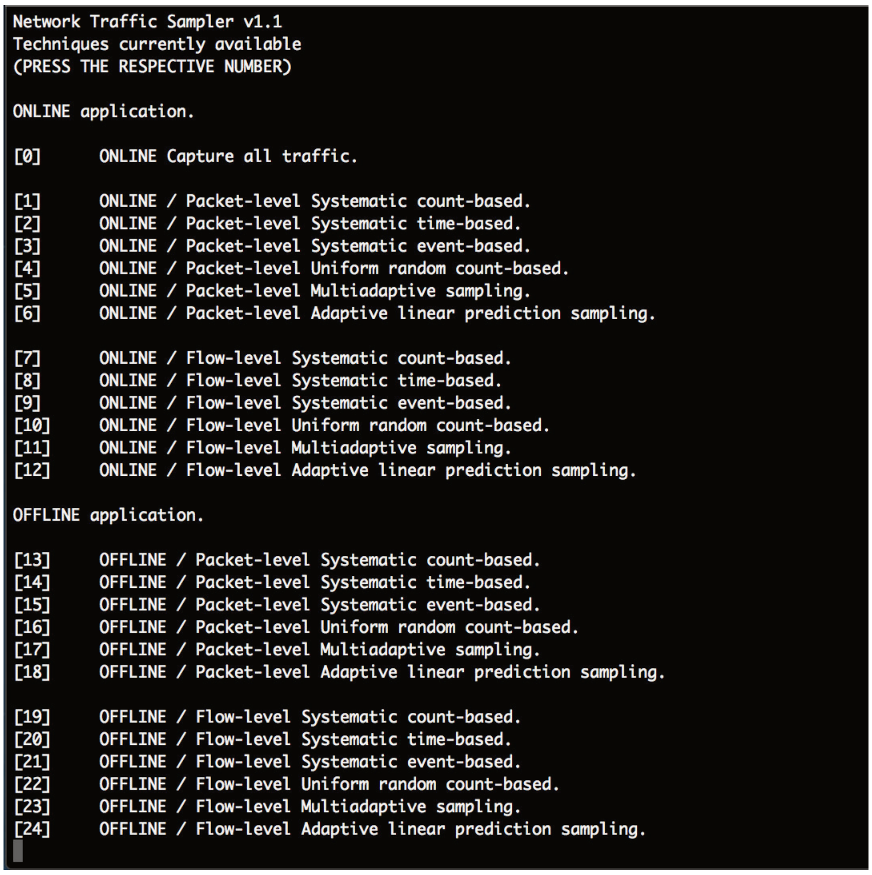
Technique	**Granularity**		**Selection trigger**			**Selection scheme**		
	Packet	Flow	Count	Time	Event	Systematic	Random	Adaptive
SystC - Systematic Count-based	✓		✓			✓		
SystC_F - Systematic Count-based_Flow-level		✓	✓			✓		
SystT - Systematic Time-based	✓			✓		✓		
SystT_F - Systematic Time-based_Flow-level		✓		✓		✓		
SystEvt - Systematic Event-based	✓				✓	✓		
SystEvt_F - Systematic Event-based_Flow-level		✓			✓	✓		
RandC - Uniform-Random Count-based	✓		✓				✓	
RandC_F - Uniform-Random Count-based_Flow-level		✓	✓				✓	
LP - Linear Prediction	✓			✓				✓
LP_F - Linear Prediction_Flow-level		✓		✓				✓
MuST - Multiadaptive Sampling Technique	✓			✓				✓
MuST_F - Multiadaptive Sampling Technique_Flow-level		✓		✓				✓

The developed framework is currently supporting research work assessing the suitability of the different sampling techniques when applied to various network measuring tasks. The comparative evaluation study takes into account the measurement accuracy, the volume of data involved and the computational weight. The corresponding results are discussed in Chapter 7.

Figures 6.2 and 6.3 present screenshots of the framework local user interface showing the sampling techniques currently available (Figure 6.2) and the list of the MP interfaces

[4]The framework and its source code are available for download at http://1drv.ms/1IggkCa as a *Raspbian* image ready to be deployed.

in which the packet sampling can be performed (Figure 6.3).



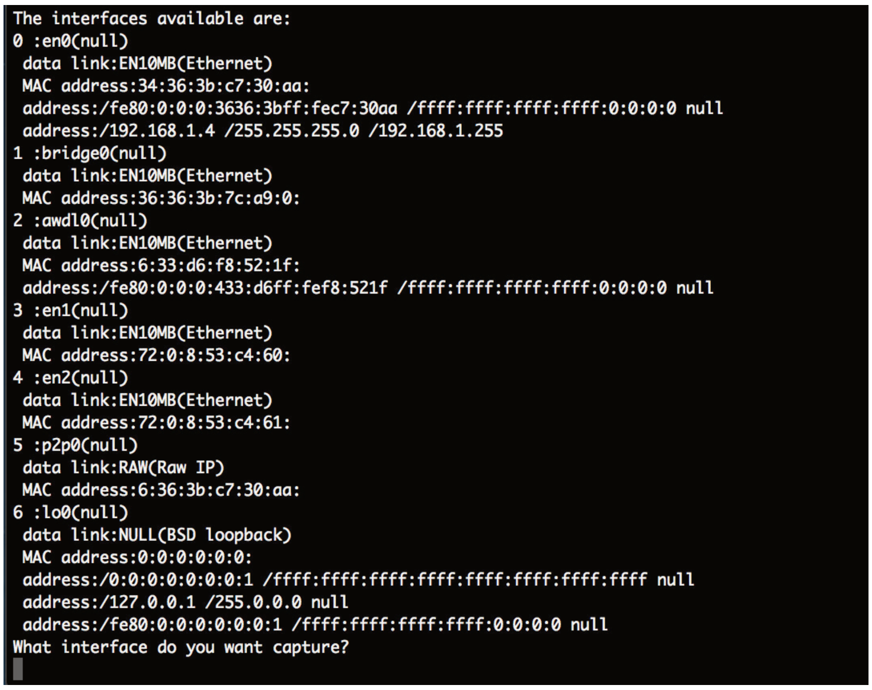
```
Network Traffic Sampler v1.1
Techniques currently available
(PRESS THE RESPECTIVE NUMBER)

ONLINE application.

[0]      ONLINE Capture all traffic.

[1]      ONLINE / Packet-level Systematic count-based.
[2]      ONLINE / Packet-level Systematic time-based.
[3]      ONLINE / Packet-level Systematic event-based.
[4]      ONLINE / Packet-level Uniform random count-based.
[5]      ONLINE / Packet-level Multiadaptive sampling.
[6]      ONLINE / Packet-level Adaptive linear prediction sampling.

[7]      ONLINE / Flow-level Systematic count-based.
[8]      ONLINE / Flow-level Systematic time-based.
[9]      ONLINE / Flow-level Systematic event-based.
[10]     ONLINE / Flow-level Uniform random count-based.
[11]     ONLINE / Flow-level Multiadaptive sampling.
[12]     ONLINE / Flow-level Adaptive linear prediction sampling.

OFFLINE application.

[13]     OFFLINE / Packet-level Systematic count-based.
[14]     OFFLINE / Packet-level Systematic time-based.
[15]     OFFLINE / Packet-level Systematic event-based.
[16]     OFFLINE / Packet-level Uniform random count-based.
[17]     OFFLINE / Packet-level Multiadaptive sampling.
[18]     OFFLINE / Packet-level Adaptive linear prediction sampling.

[19]     OFFLINE / Flow-level Systematic count-based.
[20]     OFFLINE / Flow-level Systematic time-based.
[21]     OFFLINE / Flow-level Systematic event-based.
[22]     OFFLINE / Flow-level Uniform random count-based.
[23]     OFFLINE / Flow-level Multiadaptive sampling.
[24]     OFFLINE / Flow-level Adaptive linear prediction sampling.
```

Figure 6.2: *User interface - Sampling techniques available in the framework*

Furthermore, aiming at reducing the overall costs involved in large scale network monitoring, the framework is also being explored in an experimental and distributed low cost measurement environment. In this way, the solution is deployed resorting to general purpose low-cost single-board computers, *i.e.*, Raspberry Pi Model B, running an ARM processor at 700MHz and with 512MB RAM. This type of equipment is taking ground in the network monitoring context, namely in large scale measurement architectures, such as CAIDA (Center for Applied Internet data Analysis) Archipelago Project[5].

The experimental environment is running at the *Instituto Nacional de Estatística* (INE) network and consists of nine devices attached to the INE domain border routers across the country with nominal bandwidth between 8Mbps and 500Mbps (detailed in Figure 6.4). The tasks supported by the system comprise network monitoring, accounting and traffic classification. The distributed monitoring layout allows to overcome the hardware constraints of Raspberry Pi devices for most links, since the mean throughput in these links are usually within their capabilities. Whenever an excessive throughput

[5]Details regarding the Archipelago Project can be found in http://www.caida.org/projects/ark/



```
The interfaces available are:
0 :en0(null)
 data link:EN10MB(Ethernet)
 MAC address:34:36:3b:c7:30:aa:
 address:/fe80:0:0:0:3636:3bff:fec7:30aa /ffff:ffff:ffff:ffff:0:0:0:0 null
 address:/192.168.1.4 /255.255.255.0 /192.168.1.255
1 :bridge0(null)
 data link:EN10MB(Ethernet)
 MAC address:36:36:3b:7c:a9:0:
2 :awdl0(null)
 data link:EN10MB(Ethernet)
 MAC address:6:33:d6:f8:52:1f:
 address:/fe80:0:0:0:433:d6ff:fef8:521f /ffff:ffff:ffff:ffff:0:0:0:0 null
3 :en1(null)
 data link:EN10MB(Ethernet)
 MAC address:72:0:8:53:c4:60:
4 :en2(null)
 data link:EN10MB(Ethernet)
 MAC address:72:0:8:53:c4:61:
5 :p2p0(null)
 data link:RAW(Raw IP)
 MAC address:6:36:3b:c7:30:aa:
6 :lo0(null)
 data link:NULL(BSD loopback)
 MAC address:0:0:0:0:0:0:
 address:/0:0:0:0:0:0:0:1 /ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff null
 address:/127.0.0.1 /255.0.0.0 null
 address:/fe80:0:0:0:0:0:0:1 /ffff:ffff:ffff:ffff:0:0:0:0 null
What interface do you want capture?
```

Figure 6.3: *User interface - Selecting network interface*

cannot be accommodated, the low cost involved in this monitoring strategy allows to include new devices or to split the traffic into redundant idle devices.

Due to the experimental nature of the measurement system, the collected packets are not being automatically aggregated, which allows the usage of the same capture in diverse measurement tasks. In this way, all collected packets are firstly stored in *pcap* file format for future aggregation regarding the specific measurement needs.

Regarding the exporting process, the current version of the framework provides the capability of exporting the measurement results in IPFIX format by connecting the framework with a modified version of the application *yaf*. In the tailored version of *yaf* was introduced features related to packet sampling, such as the identification of the technique in use and its operational parameters. Aiming at achieving compatibility with current monitoring applications, *yaf* was also adapted in order to support IPFIX exports into XML format. Some of the measurement outcomes from this experimental system sustain the comparative analysis detailed in Chapter 7.

6.4 A full operational example

As mentioned before, the system prototype is currently being used to support network monitoring, traffic accounting and classification at the INE network. In this way,

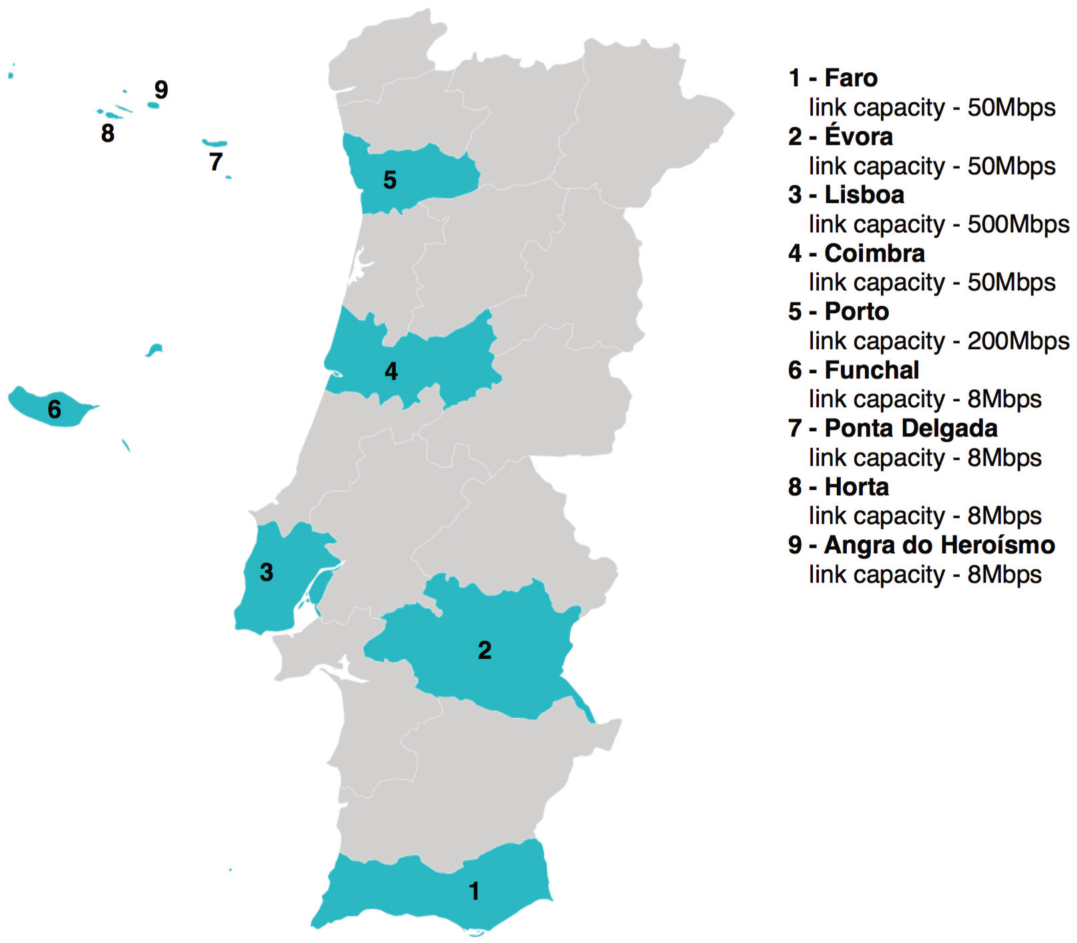


Figure 6.4: *INE experimental system*

this section illustrates the operation of the various modules presented in the measurement architecture (introduced in Chapter 5) when performing traffic accounting. The example is structured following the sequence from the network task and its underlying measurement needs to the packet selection rules being applied to the data plane. Then, the process is exemplified from the unprocessed sampled packets to the visualization of the measurement results.

This operational example includes the following steps:

1. Aiming to monitor the ISP (Internet Service Provider) compliance with the committed SLA, the INE network managers perform traffic accounting along various periods of every workday. The measurement periods are distributed in order to cover the different workloads daily supported by the network. To pursue this *network task*, the *measurement needs* involve gathering information regarding the volume of data (*i.e.*, number of packets and volume of bytes) traversing each monitored link. Thus, the main packet field of interest is the *total length* for IPv4

(Internet Protocol version 4) packets or the *payload length* for IPv6 (Internet Protocol version 6) packets;

2. Due to the nature of the task, measurements must be performed in border routers, corresponding to nine MPs attached to external links distributed along the country, as illustrated in Figure 6.4;

3. Considering the outcomes detailed in Chapter 7, the *sampling technique* selected is the Multiadaptive Sampling for links with higher capacity (*i.e.*, *Lisboa* and *Porto*) and Systematic Count-based for the remaining links. These decisions are related to the computational burden of each technique facing MPs capacity;

4. As all the MPs support the sampling framework, the elected *information model* is IPFIX due to its flexibility in supporting measurements based on sampling as in RFC 5477 [80]. As presented previously (Section 5.3.4), the information elements identifying the encoding mechanisms are ruled by IANA. Due to the comprehensive support by the monitoring applications running at INE network, the *data model* in use is XML, meaning that IPFIX messages are transmitted using XML format.

   For this example, the configuration template and underlying data messages sent from the management plane to the control plane are illustrated in Figure 6.5. Figures 6.5(a) and 6.5(b) correspond to the configuration templates for the systematic count-based technique and multiadaptive technique, respectively. Figures 6.5(c) and 6.5(d) illustrate the corresponding data messages aiming at configuring the respective technique and the exporting destination (*i.e*, 172.168.0.1). In addition, Figure 6.6 presents an example of the IPFIX data message in XML format;

5. Resorting to IPFIX messages, the control plane of each MP can *select* and *configure* the defined sampling technique. In this case, the MPs in *Lisboa* and *Porto* are configured to perform MuST with an initial sample size equal to $200ms$ and sample size equal to $500ms$. The remaining MPs are configured to perform Systematic count-base technique with fixed sample size equal to 1 and interval between samples equal to 99;

6. As described in Section 5.5, at the data plane the framework resorts to *libpcap* library to *capture packets* following the respective sampling technique policy. In order to keep this architecture plane simple, the unprocessed packets (*i.e.*, in *pcap* format) are sent to the control plane for further processing;

Version (2bytes)	Length (2bytes)
Export time (4bytes)	
Sequence number (4bytes)	
Observation domain ID (4bytes)	
Set ID = 2 (template)	Length = 32
Template ID = 321	Number of fields = 4
destinationIPv4Address (ID = 12)	
selectorAlgorithm (ID = 304)	
samplingPacketInterval (ID = 305)	
samplingPacketSpace (ID = 306)	

(a) Systematic count-based template

Version (2bytes)	Length (2bytes)
Export time (4bytes)	
Sequence number (4bytes)	
Observation domain ID (4bytes)	
Set ID = 2 (template)	Length = 32
Template ID = 325	Number of fields = 4
destinationIPv4Address (ID = 12)	
selectorAlgorithm (ID = 304)	
samplingTimeInterval (ID = 307)	
samplingTimeSpace (ID = 308)	

(b) Multiadaptive template

0x000a	48
2015-06-01 13:31:41	
0	
132436	
256	32
321	4
172.168.0.1	
1	
1	
100	

(c) Systematic count-based data

0x000a	48
2015-06-01 13:18:01	
0	
132435	
256	32
325	4
172.168.0.1	
10	
200	
500	

(d) Multiadaptive data

Figure 6.5: *IPFIX templates and data messages for measurement point configuration.*

```
<ipfix>
  <version>0x000a</version>
  <messageLength>40</messageLength>
  <exportTime>2015-08-01 21:25:02</exportTime>
  <sequenceNumber>0</sequenceNumber>
  <observationDomain>132435</observationDomain>
  <setID>256</setID>
  <payloadLength>24</payloadLength>
  <templateID>321</templateID>
  <numberOfFields>3</numberOfFields>
  <samplingTechnique elementID="304" name="selectorAlgorithm" id="1">
    <sampleSize elementID="305" name="samplingPacketInterval">1</sampleSize>
    <intervalBetweenSamples elementID="306" name="samplingPacketSpace">100</intervalBetweenSamples>
  </samplingTechnique>
</ipfix>
```

Figure 6.6: *IPFIX data message in XML format.*

7. Although the current version of the framework does not support error verification, at this point each received packet is processed in order to detect errors. The *packet processing* module also gathers the packet fields of interest for the specific measurement goal, discarding all the remaining data. Relevant data is then aggregated or directly transferred to the exporting module;

8. The process of data *aggregation* resorts to dynamic tables (also called cache tables)

updated everytime a new packet is processed. Considering the requirements for traffic accounting, the underlying table stores the information presented in Figure 6.7. The table maintains the timestamps[6] of the first and the last packet captured, the total number of packets captured in this period and the corresponding total number of bytes. Note that the timestamp considered corresponds to the time of packet capture by *libpcap*;

time stamp of first packet	time stamp of the last packet	number of packets	volume of data
1444234024.583388000	1444234028.008043000	5	3950

Figure 6.7: *Aggregation table.*

9. At the *exporting* module, the measured data (aggregated or not) is encapsulated into IPFIX messages for transmission to the management plane. Figure 6.8(a) illustrates the template for measurement exporting and Figure 6.8(b) illustrates the correspondent data message. This message carries the aggregated information related to the MP in *Evora* (*i.e.*, ID = 7), performing the Systematic count-based technique, in which 1 packet is captured for each 100 incoming packets;

Version (2bytes)	Length (2bytes)
Export time (4bytes)	
Sequence number (4bytes)	
Observation domain ID (4bytes)	
Set ID = 2 (template)	Length = 56
Template ID = 320	Number of fields = 7
selectorID (ID = 302)	
selectorAlgorithm (ID = 304)	
samplingPacketInterval (ID = 305)	
samplingPacketSpace (ID = 306)	
observationTimeSeconds (ID = 323)	
packetDeltaCount (ID = 32)	
octetDeltaCount (ID = 1)	

(a) Exporting template

0x000a	72
2015-09-02 20:12:10	
0	
132435	
250	56
320	7
7	
1	
1	
100	
3	
5	
3950	

(b) Exporting data

Figure 6.8: *IPFIX templates and data messages for measurement exporting.*

10. Resorting to the measured data embedded in IPFIX data messages, it is possible estimating the metrics related to traffic accounting (discussed in Section 7.4). In this experimental environment, the *measurement results* are stored in a database for further verification of SLA compliance. However, they can also be assessed through any visual application compatible with the IPFIX protocol.

[6]Using the *epoch time*, defined as the number of seconds elapsed since 00:00:00 Coordinated Universal Time (UTC), 1 January 1970. The precision also can be increased to the microseconds scale.

6.5 Summary

This chapter has detailed the deployment of a sampling framework based on the modular architecture previously introduced. The framework aims at providing a functional tool to support the design of efficient measurement strategies. Aspects regarding the computer architectures supported, libraries used in its deployment and the software structure were also discussed, detailing the implementation of each component presented in the sampling taxonomy. In addition, an experimental environment targeting a large scale network was presented, including a concrete example in which it is possible to track each module of the measurement architecture and their relations. The next chapter resorts to this framework to provide the proof-of-concept of the proposed architecture regarding the main objectives of this work.

Chapter 7

Test Scenarios and Results

Deciding for a specific sampling strategy requires a clear knowledge of each eligible sampling technique capabilities and constraints regarding the measurement activities to be accomplished. This includes understanding their effectiveness in providing accurate estimations and assessing the computational burden involved in achieving such accuracy.

In order to overcome the lack of current commercial and research tools in providing an encompassing environment for comparative sampling analysis, the framework presented in Chapter 6 has introduced the flexibility required for such purpose.

Thus, in this chapter, after detailing the main objectives and the general methodology adopted, multiple testing scenarios are devised in order to evaluate relevant and heterogeneous sampling techniques. The debate focuses mostly on the comparative evaluation of the overhead in terms of volume of data and computational requirements and the accuracy in estimating the traffic workload and flow distribution. Relevant aspects regarding the challenges in performing network monitoring though packet sampling are also covered.

7.1 Main objectives

The main objectives of evaluating different sampling techniques in presence of diverse traffic scenarios are twofold. In a first stage, the aim is to demonstrate the versatility of the proposed sampling taxonomy in providing a flexible solution to accommodate diverse sampling strategies and, therefore, fostering the development of tailored schemes. In a second stage, the sampling framework is used as a fair environment for evaluating the performance of the different sampling techniques when addressing activities usually supported by sampled data, *i.e.*, *traffic workload* and *flow analysis*. The performance is assessed comparing estimation accuracy and computational requirements when applying the distinct techniques to the mentioned activities.

In order to meet these objectives, specific tasks are identified and listed below:

- deploy a set of sampling techniques comprising: (*i*) the most common techniques currently available in research or commercial tools; (*ii*) standard techniques scarcely offered by these tools; and (*ii*) techniques recently proposed;
- select traffic scenarios able to represent the real environment in which traffic sampling is usually performed;
- conceive and implement a test environment able to fairly assess the computational burden of the different sampling techniques. The computational resources measured consist of *CPU load*, *memory usage* and *volume of data* involved in each sampling process;
- identify metrics commonly used in traffic measurements and deploy the underlying estimation methods taking the sampled data inputs. The metrics are mostly related to traffic workload and flow analysis, however several highlights regarding different measurement tasks are also covered along the discussion.

7.2 General methodology of tests

Some aspects regarding the methodology used to accomplish the objectives defined above are kept unchanged along all the test scenarios. These aspects are presented in this section, while specific details (*i.e.*, traffic scenarios and evaluation methods) concerning each set of tests are presented in the respective sections.

The main unchanged aspect in the tests regards to the sampling techniques evaluated, which were chosen in order to sustain a comprehensive analysis of the different sampling mechanisms and their impact on metrics estimation.

In this way, the techniques evaluated include the main strategies defined in RFC5475 [1]. Two of them are widely deployed in current measurement tools, *i.e.*, *systematic count-based* and *uniform random count-based*, while the other, *i.e.*, *systematic time-based*, is scarcely deployed. In addition, two adaptive techniques are also evaluated, *i.e.*, *adaptive linear prediction* [28] and *multiadaptive sampling* [14]. Relevant details regarding each sampling technique are presented below and a summary of their nomenclature and operational parameters is presented in Table 7.1.

Systematic count-based (SystC) drives the packet selection through a deterministic and invariable function based on the packet position, using counters. This technique, proposed in RFC5475 [1], corresponds to the most widely deployed in current measurement tools able to perform packet sampling, such as *Cisco Sampled Netflow*, *sFlow* and *tcpdump*.

Considering that the estimation accuracy and the computational resources consumption of systematic techniques are empirically proportional to the sampling frequency, the tests carried out assess the impact of distinct sampling frequencies. As it is exemplified in Table 7.1, SystC 1/8 means that one packet is collected over every eighth packets arriving at the MP. The configuration of SystC used in comparative analysis among different techniques is 1/100, as suggested in [33].

Systematic time-based (SystT) the process of packet selection follows a deterministic function based on the packet arrival time at the MP. In this technique the sample size and the time between samples are set at the beginning of measurements and remain unchanged along the sampling process. Although this technique is also proposed in RFC 5475 [1], it is scarcely available in current measurement tools.

Similarly to SystC technique, SystT is also analyzed on several frequencies (presented in Table 7.1), where SystT 100/500 means that all packets arriving at the MP along a period of 100ms are selected for a sample, followed by a time period of 400ms where packets are ignored for measurement purposes. The chosen parameters also allow analysing the variation of SystT frequencies with the same sampling ratio, *i.e.*, 100/500, 200/1000 and 500/2500.

For SystT technique, the default sampling frequency when comparing the performance of different techniques is 100/1000 as it led to the best results for the analysis performed.

Random count-based (RandC) corresponds to the *n-out-of-N* random approach introduced in RFC 5475 [1], widely deployed in measurement tools. It is a content-independent technique in which n packets are randomly selected out of a parent population of size N, with all N packets having the same probability to be selected for the sample.

Following the suggestion in [33], when comparing RandC and with other techniques, a frequency of 1/100 is used, in which one packet is randomly selected from every 100 incoming packets at the MP. In other words each packet has a 1/100 probability to be selected.

Adaptive linear prediction (LP) is a time-based technique that uses linear prediction to identify changes in the network activity, adjusting the sampling frequency accordingly while the sample size remains invariant [28]. Its basic operation consists of increasing the sampling frequency, *i.e.*, reducing the interval between samples, when more network activity than predicted is observed. Conversely, when less network activity than predicted is observed, the interval between samples is increased, reducing the sampling frequency and, consequently, the amount of data involved in the sample process.

Within the comparative context, LP technique is configured with sample size equal to 100ms and initial interval between samples equal to 200ms, as suggested in [28].

Multiadaptive sampling (MuST) is a time-based technique (introduced in Chapter 4) that also resorts to linear prediction for identifying the level of network activity. However, the multiadaptive technique considers both the interval between samples and the sample size as adjustable parameters [14].

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 MP in a critical period. 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 MP.

Due to the dynamic behavior of the sample size and interval between samples, the operational parameters only rule the initial samples. In this case, the initial parameters are set to 200ms and 500ms for the sample size and interval between samples, respectively.

The general methodology of tests consists in applying the different sampling techniques to real traffic scenarios in order to understand how the strategies behave considering the same conditions and evaluation criteria. To pursue this, all techniques are submitted to real traffic traces previously collected representing distinct and relevant traffic scenarios of operational networks. Thereafter, specific metrics are estimated and statistically compared. Although the statistical parameters used are common in most representative research works on packet sampling, this work extends the previous ones by cross-checking results from sampling and total traffic (*i.e.*, unsampled) and by extending the comparison to other techniques than systematic count-based.

Table 7.1: *Summary of the evaluated sampling techniques and corresponding parameters*

Name	**Description**	**Sampling frequency**
SystC	Systematic count-based	1 out of every 100 packets
SystC 1/8	Systematic count-based	1 out of every 8 packets
SystC 1/16	Systematic count-based	1 out of every 16 packets
SystC 1/32	Systematic count-based	1 out of every 32 packets
SystC 1/64	Systematic count-based	1 out of every 64 packets
SystC 1/128	Systematic count-based	1 out of every 128 packets
SystC 1/256	Systematic count-based	1 out of every 256 packets
SystT	Systematic time-based	100ms out of 1000ms
SystT 100/500	Systematic time-based	100ms out of 500ms
SystT 200/500	Systematic time-based	200ms out of 500ms
SystT 200/1000	Systematic time-based	200ms out of 1000ms
SystT 500/1500	Systematic time-based	500ms out of 1500ms
SystT 500/2500	Systematic time-based	500ms out of 2500ms
SystT 500/3500	Systematic time-based	500ms out of 3500ms
RandC	Uniform random count-based	1 packet out of every 100 packets
LP	Time-based adaptive linear prediction	fixed sample size equal to 100ms and initial interval between samples equal to 200ms
MuST	Time-based multiadaptive	initial sample size equal to 200ms and initial interval between samples equal to 500ms

The following sections evaluate the sampling techniques regarding: (*i*) the computational weight related to the CPU load, memory usage and volume of data involved in measurements; (*ii*) accuracy in estimating traffic workload; and (*iii*) accuracy in analyzing traffic flows. Each scenario and the specific methodology of tests are detailed along each section.

7.3 Evaluation of the computational weight

The computational weight of packet sampling techniques is analyzed regarding the *CPU load*, the *memory consumption* and the *volume of data* involved in the sampling process. For this, a set of real traffic traces with different loads is applied to the defined packet sampling techniques, deployed in the same device.

7.3.1 Methodology of tests

To perform the tests, the sampling framework presented in Section 6.3 was installed in a general purpose low-cost single-board computer, *i.e.*, Raspberry Pi Model B, running an ARM processor at 700MHz and 512MB RAM. The selected traces (detailed below) are injected into this MP executing each technique individually, ensuring that all techniques are evaluated under the same conditions.

Traffic scenarios

The traffic scenarios used to evaluate the computational weight correspond to three workload periods (*low*, *moderate* and *high*) in the network backbone of the University of Minho (UMinho) campus along a typical workday. Each trace corresponds to a ten minutes long capture, classified considering the *Mean Throughput* (MTP) regarding the capacity of the network interface, *i.e.*, 100Mbps. Due to institution policies, only *https (Hyper Text Transfer Protocol Secure)* traffic was collected, allowing to keep the privacy of users data.

For the analysis of the volume of data involved in each sampling technique, traces captured in a large Internet Service Provider (ISP) from the US West Coast, publicly available at CAIDA are also used. These traces correspond to captures of approximately 5 minutes in a backbone link OC48 [61], which has a nominal throughput of approximately 2.5 Gbps, and OC192 [104], with nominal throughput of approximately 10Gbps.

A summary of all traffic scenarios are presented in Table 7.2.

Table 7.2: *Traffic scenarios*

Workload scenario / Feature	Description	Number of packets	Volume of data (MBytes)	Mean throughput (Mbps)
Low	UMinho network	311159	112.04	3.68
Moderate	UMinho network	1273068	712.99	25.42
High	UMinho network	1718804	1063.16	65.61
OC48	CAIDA [61]	13575655	7320.62	975.97
OC192	CAIDA [104]	15335726	11585.69	1546.78

Comparative parameters

The computational weight of each sampling technique is analyzed in terms of hardware resources usage and volume of sampling data stored (without aggregation). While resource usage may impact on the performance of the MP, data volume affects the bandwidth required to export measurement data as well as the storage and processing overhead [71].

The computational resources observed along the sampling process are:

- *CPU load*, corresponds to the time spent running non-kernel code measured using *vmstat*[1] [105]. This means that only the sampling processes are considered;
- *Memory usage*, also measured through *vmstat*, corresponds to the amount of active memory in the MP. In this case, the analysis considers all memory used by the MP due to the process of copying the selected packets from the kernel space to the user space;
- *Volume of data*, corresponds to the sum of all packets collected by each sampling technique, using the *total length* field within IP header;
- Relation between the *volume of data* collected and the *computational burden* involved, assessed through $\frac{\%CPU}{MByte}$ and $\frac{\%Memory}{MByte}$.

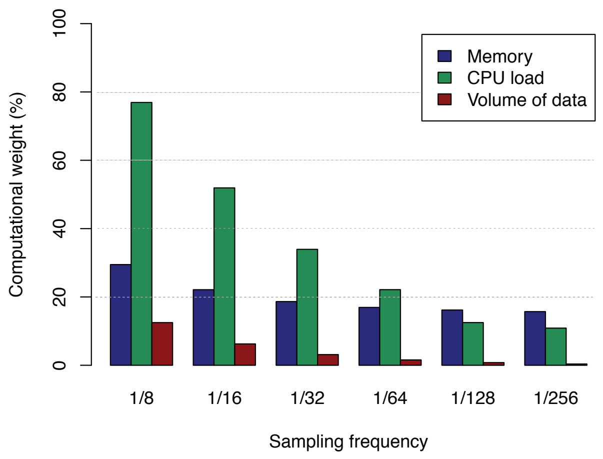
7.3.2 Evaluating systematic techniques

Considering that the computational resources consumption of systematic techniques is intuitively proportional to the sampling frequency (as higher sampling frequencies lead to more packets captured), Figure 7.1 shows the difference of the mean computational consumption for distinct sampling frequencies when applied to the high workload scenario.

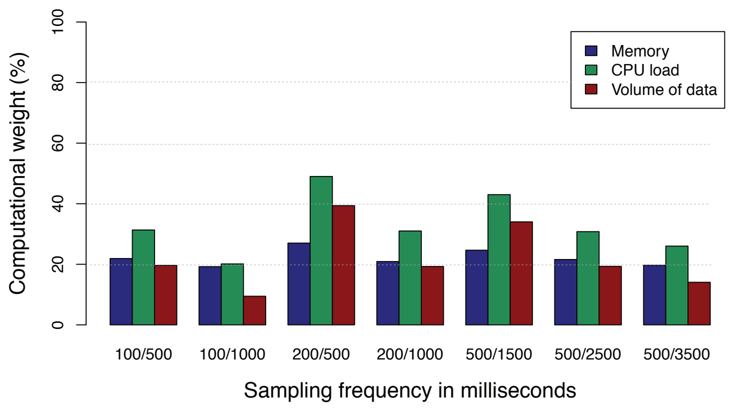
Essentially, Figure 7.1 confirms the relation between sampling frequency and computational weight, although the CPU load presents a higher variation than memory usage across the different frequencies for both systematic approaches. In more detail, SystC technique (Figure 7.1(a)) exhibits a smoother variation of computational weight as the sampling frequency decreases, suggesting a stable minimum demand in terms of resources for low frequencies.

For SystT (Figure 7.1(b)), besides the relation with the sampling frequency, the computational weight varies for different frequencies with the same sampling ratio, *i.e.*, 100/500, 200/1000 and 500/2500. This occurs as the stochastic behavior of the traffic varies the proportion of the traffic that will be collected through the sampling.

[1]A computer monitoring tool able to collect information about the operating system activity on a near real-time basis [105].



(a) SystC



(b) SystT

Figure 7.1: *Systematic techniques comparison - High workload*

Therefore, as depicted in Figure 7.1(b), SystT leads to distinct volume of data collected, and consequently different computational requirements by each sampling frequency, even with the same temporal ratio.

In addition, Figure 7.1 also demonstrates the relation between sampling frequency and volume of data processed and stored by the systematic techniques. As observed for CPU and memory consumption, SystC presents a proportional relation between the frequency and data amount. However, although SystC is more widely used in current measurement tools than SystT, the second one requires less computational resources when considering an equivalent volume of sampled data. For instance, the sampling frequencies of SystC 1/8 and SystT 500/3500, collect around 30% of all traffic in the high workload scenario. In this case, the SystC requires, in average, around 30% of CPU and 77% of memory, while the SystT requires around 25% of CPU and 43% of memory. Furthermore, this behavior is also observed for all traffic scenarios.

Table 7.3 extends the results of the mean resource consumption of systematic techniques for all traffic scenarios, confirming the higher variance of CPU load when facing memory consumption for all techniques and scenarios. Moreover, it is possible to observe a stabilization of memory consumption (around 15%) for SystC in lower frequencies. This indicates the lower demand of this technique in Raspberry Pi MPs. Analysing the MP active processes, it is also observed that the higher variation in the CPU load is mainly related to the operational system process of copying more selected packets from the kernel space to the user space.

Table 7.3: *Systematic technique comparison - computational weight*

Frequency	**High**		**Moderate**		**Low**	
	Memory	CPU	Memory	CPU	Memory	CPU
SystC 1/8	29.48%	76.92%	26.33%	42.47%	17.05%	11.89%
SystC 1/16	22.12%	51.92%	20.45%	27.80%	16.16%	6.1%
SystC 1/32	18.65%	33.93%	17.86%	16.47%	15.73%	5.53%
SystC 1/64	16.93%	22.14%	16.66%	12.66%	15.53%	5.21%
SystC 1/128	16.18%	12.50%	15.99%	8.25%	15.28%	2.75%
SystC 1/256	15.71%	10.88%	15.67%	5.52%	15.21%	2.57%
SystT 100/500	21.90%	31.33%	22.71%	28.05%	22.30%	26.89%
SystT 100/1000	19.22%	20.12%	19.36%	17.95%	19.26%	14.55%
SystT 200/500	27.00%	49.00%	30.85%	47.33%	26.46%	38.60%
SystT 200/1000	20.91%	31.00%	23.34%	27.27%	22.41%	23.82%
SystT 500/1500	24.65%	43.00%	22.92%	40.82%	26.10%	33.21%
SystT 500/2500	21.58%	30.78%	22.30%	28.80%	22.83%	21.71%
SystT 500/3500	19.67%	26.00%	21.50%	20.27%	20.77%	18.42%

7.3.3 Comparing different techniques

Regarding the comparative analysis of all sampling techniques, Figure 7.2 presents the evolution of CPU load along the sampling process for the high workload scenario. As shown, the LP technique clearly demands more CPU resources than the other techniques. This occurs because this technique requires processing all packets to analyze the variation of traffic activity based on accumulated data, even packets not collected. Conversely, the MuST technique requires the lowest CPU usage, confirming its main goal and ability to reduce the resource consumption during high activity periods [14]. This aspect is particularly relevant for high-load, high-speed networks.

The complete comparison of the average CPU load is presented in Table 7.4. As shown, SystC and RandC techniques outperform MuST for low workload scenarios. This is due to the impact of the adaptive process on self-adjusting facing workload variations and also to the largest number of packets collected by MuST in periods of low activity. In addition, the difference in CPU load of count-based techniques (SystC and RandC), for the same sampling frequency, results from the additional cost of the random function in RandC, suggesting that the more complex the random function is the more significant this difference will be.

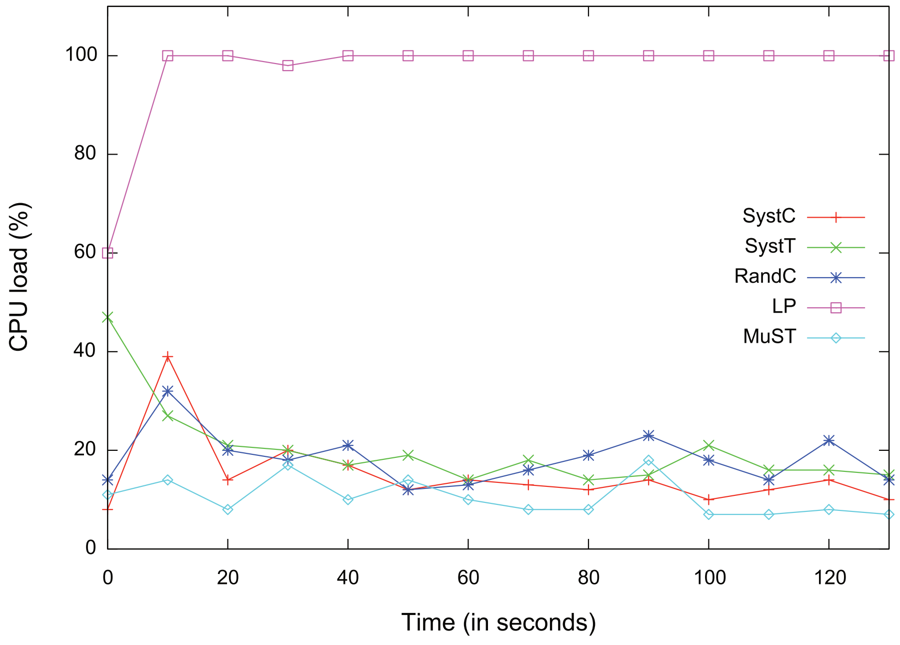


Figure 7.2: *CPU load - High workload*

Figure 7.3 illustrates the memory usage by each sampling technique along the sampling process of the high workload scenario, in which the SystT technique requires the highest amount of memory from the MP. This is also observed for all traffic scenarios considered, as detailed in Table 7.4, and it is directly related to the highest number of captured packets and consequently the highest volume of data collected by this technique.

For the high workload scenario (detailed in Figure 7.3), the SystC technique demands the lowest memory amount, although none technique has incurred in a significant resource consumption, as observed in the CPU load analysis for the LP technique. Regarding MuST, Table 7.4 also ratifies its ability to reduce the memory consumption during high activity periods, as previously observed in the CPU load analysis.

Regarding the volume of data collected and stored along the sampling process for

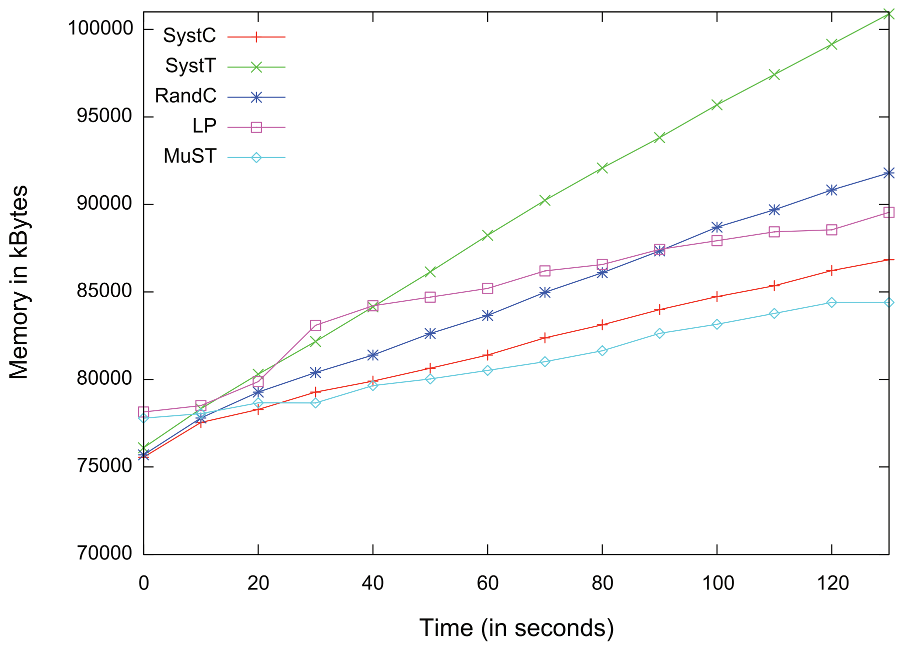


Figure 7.3: *Memory usage - High workload*

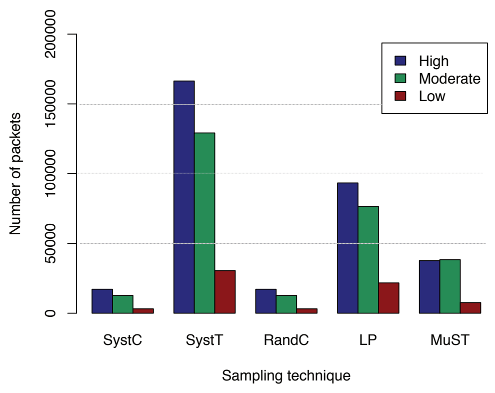
Table 7.4: *Average use of computational resources - all traffic scenarios*

Technique	**High**		**Moderate**		**Low**	
	Memory	CPU	Memory	CPU	Memory	CPU
SystC	16.42%	14.92%	16.22%	10.80%	15.37%	5.03%
SystT	18.22%	20.12%	19.36%	17.95%	19.26%	14.55%
RandC	17.30%	18.26%	16.87%	16.86%	16.31%	5.50%
LP	17.18%	97.27%	17.61%	96.68%	16.55%	27.35%
MuST	16.22%	10.76%	16.94%	10.72%	17.13%	8.82%

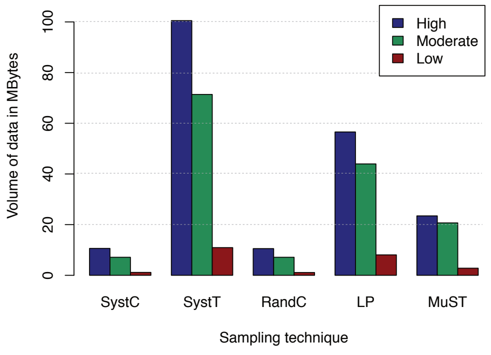
the three workload scenarios, the graphics in Figure 7.4 illustrate the total number of packets collected by each technique (Figure 7.4(a)) and the corresponding volume of data in MBytes (Figure 7.4(b)). As shown, the count-based techniques use less resources when comparing to all time-based techniques under analysis. At this point, it is important to observe that count-based techniques allow a previous definition of the proportion of the total traffic that will be collected (in number of packets). This makes these techniques more suitable for MPs with storage limitations or for reducing the impact of measurement data traversing the network during the sampling exporting process.

For time-based techniques, MuST also achieves the best results in the data volume analysis. This also confirms the performance improvement in reducing resource requirements for all traffic scenarios, as illustrated in Figure 7.4.

Although the previous analysis yields a fair comparison among different techniques



(a) Number of packets



(b) Volume of data

Figure 7.4: *Comparative data volume*

regarding their computational requirements, it is also important to understand how the consumption of resources is related among techniques. In this way, Table 7.5 extends the previous analysis by assessing the ratio of CPU load per Mbyte collected and memory consumption per Mbyte collected.

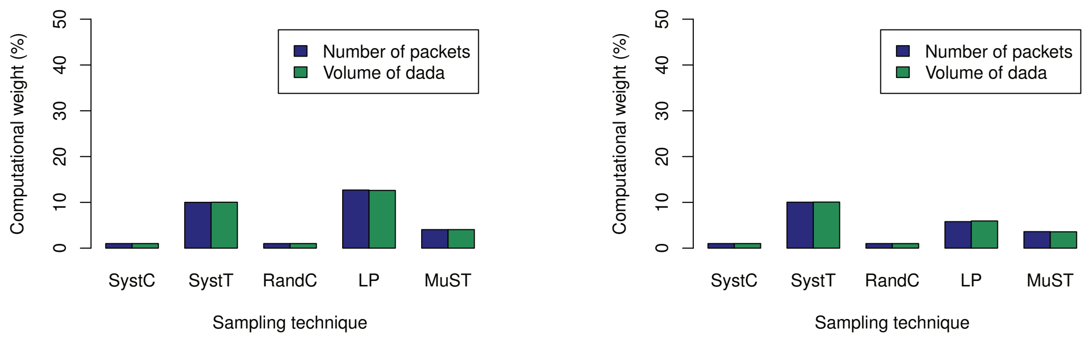
As presented in Table 7.5, although with higher consumption of storage resources, SysT and MuST techniques achieve a better relationship between the volume of data collected and the computational cost involved. This means that these techniques are more efficient regarding the computational burden. This may be an advantage for network activities in which more information about the traffic improves the results in measurement accuracy (*e.g.*, traffic classification and intrusion detection). This relation can also help on defining the most demanding technique, a concept used to decide on a sampling strategy (discussed in Section 5.3.2).

Table 7.5: *Computational resource per Mbyte collected*

Technique	**High**		**Moderate**		**Low**	
	$\frac{\%Memory}{MByte}$	$\frac{\%CPU}{MByte}$	$\frac{\%Memory}{MByte}$	$\frac{\%CPU}{MByte}$	$\frac{\%Memory}{MByte}$	$\frac{\%CPU}{MByte}$
SystC	1.63	1.40	2.52	1.51	13.36	4.37
SystT	0.16	0.20	0.27	0.25	1.76	1.33
RandC	1.73	1.73	2.34	2.36	14.82	5.00
LP	0.32	1.71	0.40	2.24	2.05	3.38
MuST	0.72	0.45	0.81	0.48	4.59	2.36

Aiming at extending the comparative discussion regarding the data amount involved in each sampling technique, Figure 7.5 presents the percentage (regarding the unsampled trace) of the number of packets and volume of data for traces captured in high capacity links, *i.e.*, OC48 in Figure 7.5(a) and OC192 in Figure 7.5(b).

Taking into account the unpredictable sampled data in time-based approaches, MuST has remained the most efficient technique regarding this selection trigger group. The results also show a close relation between the two parameters (*i.e.*, number of packets and volume of data) for all traffic scenarios and sampling techniques, in which the percentage of packets collected corresponds approximately to the percentage of Mbytes collected. For instance, in trace OC48, SystC has captured 1% of the original packets corresponding to 1.2% of the total data amount in Mbytes.



(a) OC 48 (b) OC 192

Figure 7.5: *Comparative data volume - traces from CAIDA*

The outcome from the computational weight analysis has demonstrated that, despite the extensive deployment of count-based techniques in current measurement tools, time-based techniques SystT and MuST achieve a better relationship between volume of sampled data and computational resource usage. In the overall packet sampling process, the CPU is a more demanded resource than memory, also exhibiting a higher consumption variation. Considering the volume of data involved in sampling and consequently in exporting, among the time-based techniques, MuST achieves better results regardless of the traffic scenario.

Although the present study considers a specific test environment regarding the MP capabilities, the obtained results bring a valuable comparative insight among existing sampling techniques. The present contribution is therefore a step forward in providing a better understanding the overhead associated with traffic sampling techniques regarding their deployment in real scenarios, helping in the adoption of more efficient measurement strategies.

7.4 Estimating traffic workload

Despite the importance of reducing the consumption of computational resources associated with packet sampling, a sampling technique must still be able to represent the network behavior accurately. Several monitoring activities and management decisions are performed resorting to the measurement of the traffic behavior, usually observing the link utilization. Thus, this section analyzes and compares each sampling technique ability in providing accurate estimations regarding the traffic workload, which consists in the most common parameter to assess network behavior.

7.4.1 Methodology of tests

The methodology adopted in these comparative tests consists in submitting real traffic from different network scenarios to all sampling techniques and then assessing metrics related to traffic workload, *e.g.*, *mean throughput* and *instantaneous throughput*. This comparative approach considers the accuracy of the sampling estimations regarding the unsampled traffic, instead of comparing just one sampling technique with the others.

Traffic scenarios

Aiming at complementing the computational weight evaluation with the analysis of the correctness in estimating traffic behavior, the first comparative assessment uses the same traffic scenarios of previous tests (*i.e.*, captures from the University of Minho network in *low*, *moderate* and *high* workload scenarios).

The public traces from ISP backbones (*i.e.*, OC48 [61] and OC192 [104]) available at CAIDA are also used to extend the discussion to high-speed networks.

A summary of the used traffic scenarios is presented in Table 7.6.

Table 7.6: *Traffic scenarios*

Traffic scenario	Description	Mean throughput (Mbps)	Mean packet size (Bytes)
Low	UMinho network	3.68	377.58
Moderate	UMinho network	25.42	587.26
High	UMinho network	65.61	648.59
OC48	CAIDA [61]	975.97	565.44
OC192	CAIDA [104]	1546.78	792.16

Comparative parameters

In this assessment, each sampling technique is analyzed regarding its ability in producing measurement data able to represent the overall traffic behavior. A common way to achieve this goal is estimating throughput, measuring the amount of sampled data in a time interval. Thereby, the accuracy in estimating the temporal traffic behavior is analyzed through *instantaneous throughput*, *i.e.*, the throughput constantly estimated along the measurement process, and *mean throughput*, *i.e.*, the total estimated load during the full sampling process.

Considering that in traffic sampling only a subset of total network packets is captured and considered for measurement purposes, estimating the traffic throughput must consider the unselected packets. The most common method to estimate the mean throughput from sampled data resorts to the statistical extrapolation based on the proportional number of unsampled packets, as detailed in Equation 7.1 [71].

$$\bar{X} = \frac{(\sum_{i=1}^{n} X_i) * S_p}{\Delta T} \tag{7.1}$$

where,

$\bar{X}$ is the estimated mean throughput;
X_i is the size of the ith sampled packet;
S_p is the statistical sampling proportion defined by $S_p = \frac{m}{n}$, with m as the total number of arriving packets and n the total number of sampled packets;
ΔT is the period of observation in seconds.

Following Equation 7.1, the mean throughput is estimated taking ΔT equal to the total period of the sampling process, while in the instantaneous throughput ΔT is defined according to the time interval of each sample[2]. This information can be obtained from the exported measurement data into IPFIX messages and may be accumulated due the aggregation process.

The accuracy in estimating traffic workload is assessed through Relative Mean Error (RME), which expresses the relative error of the estimated mean throughput regarding the mean throughput of the unsampled traffic, as detailed in Equation 7.2 [46].

$$RME = \frac{|\bar{X_u} - \bar{X_e}|}{\bar{X_u}} \tag{7.2}$$

[2]For count-based techniques with sample size equal to 1, the instantaneous throughput considers ΔT equal to $100ms$.

where,

$\bar{X}_u$ is the mean throughput of the unsampled traffic;
$\bar{X}_e$ is the estimated mean throughput after each sampling process.

Furthermore, the *mean packet size* and complementary descriptive statistics to measure the variability of packet time series, *i.e.*, the *ratio between peak and average* packet size, are also analyzed.

Understanding the sampling impact on these workload dimensions (*i.e.*, mean and instantaneous throughput) aims to provide inputs in different time scales activities. Capacity planning and topology changes, upgrade of new routers and links, and evaluation of network and protocols design may resort to measurements along periods from hours to months. Conversely, monitoring DDoS attacks, serious congestion and routers or link failures require measurements performed near real-time.

7.4.2 Evaluating systematic techniques

Similarly to the computational weight evaluation, systematic techniques were firstly analyzed considering different sampling frequencies. Table 7.7 presents the estimated mean throughput of the three workload scenarios from the University of Minho network and the two backbone traces from CAIDA.

The results in Table 7.7 show that, despite the significant variation in the computational requirements across the various frequencies of SystC, the estimated mean throughput does not vary significantly, as demonstrated by the low and similar RME. These results suggest that, for activities such as accounting, saving resources by collecting less packets does not lead to less accurate measurements. For OC48 and OC192 traces, the reduction in stored and transmitted data can be even more significative.

Changing the frequency in SystT leads to a more variable accuracy in mean throughput estimations, as demonstrated by the variation of RME in Table 7.7. In this case, unlike SystC, reducing the sampling frequency of SystT implies less accurate estimations. This is mainly due to the higher interval in which no packets are collected from the network, missing thus, significative periods of network activity.

Pondering SystC and SystT results, despite the overall higher RME in SystT estimations, the difference between their computational burden allied to the measurement goals can be decisive in choosing one of them and their respective frequency. A relevant situation in which this balance may be crucial is presented in Section 7.5, where

Table 7.7: *Systematic technique comparison - mean throughput*

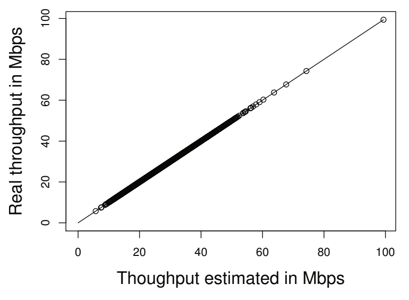
Frequency	**High**		**Moderate**		**Low**		**OC48**		**OC192**	
	Mbps	RME	Mbps	RME	Mbps	RME	Mbps	RME	Mbps	RME
Unsampled	65.61	-	25.42	-	3.68	-	975.97	-	1534.78	-
SystC 1/8	65.60	0.0002	25.38	0.0013	3.69	0.0028	976.40	0.0004	1534.97	0.0001
SystC 1/16	65.69	0.0011	25.57	0.0059	3.71	0.0070	976.84	0.0009	1535.13	0.0002
SystC 1/32	65.73	0.0018	25.55	0.0049	3.69	0.0028	975.69	0.0002	1536.26	0.0009
SystC 1/64	65.99	0.0057	25.51	0.0034	3.69	0.0028	975.34	0.0006	1535.29	0.0003
SystC 1/128	65.41	0.0030	25.60	0.0072	3.69	0.0028	980.65	0.0048	1535.91	0.0007
SystC 1/256	65.24	0.0056	25.56	0.0053	3.88	0.0540	979.46	0.0035	1535.69	0.0005
SystT 100/500	64.27	0.0205	25.09	0.0128	3.52	0.0429	975.56	0.0004	1538.01	0.0021
SystT 100/1000	63.07	0.0388	24.65	0.0302	3.51	0.0460	978.43	0.0025	1534.52	0.0001
SystT 200/500	64.57	0.0159	25.31	0.0044	3.57	0.0305	975.72	0.0002	1535.31	0.0003
SystT 200/1000	64.13	0.0226	25.05	0.0146	3.54	0.0374	977.08	0.0011	1530.21	0.0029
SystT 500/1500	62.51	0.0473	22.19	0.1270	3.51	0.0450	1100.52	0.1276	1447.66	0.0567
SystT 500/2500	70.33	0.0718	20.53	0.1921	3.19	0.1326	1183.89	0.2130	1443.91	0.0592
SystT 500/3500	53.25	0.1884	18.88	0.2571	3.22	0.1247	1099.46	0.1265	1762.35	0.1482

sampling techniques are evaluated regarding their accuracy in flow measurements.

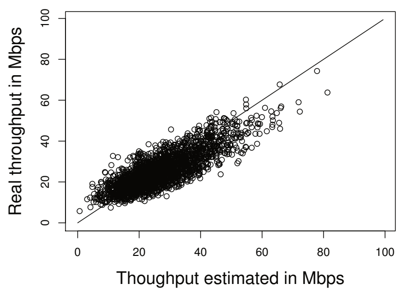
Although varying the frequency of systematic techniques does not lead to significant variation in the mean throughput accuracy, when considering the instantaneous throughput, the difference becomes more noticeable. Figure 7.6 presents the dispersion of the SystC estimations for the high workload scenario, where each point corresponds to an $100ms$ observation period and the values closer to the reference line indicate a lower estimation error. The graphics show that increasing the sampling frequency from SystC 1/8 (Figure 7.6(a)) to SystC 1/256 (Figure 7.6(c)) produce more inaccurate instantaneous throughput estimations. This is also observed for all traffic scenarios and demonstrates that the need for long term or short term workload assessments, must also guide the sampling frequency decision.

7.4.3 Evaluating different techniques

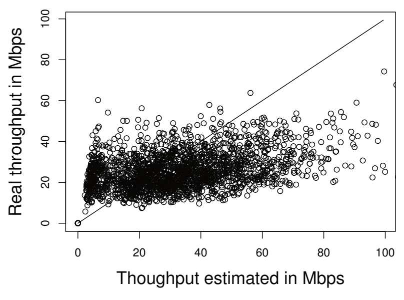
Considering the comparative analysis of different techniques, Table 7.8 presents the mean throughput estimated after each sampling process. In general, for all traffic scenarios, the RME is low (less than 5%). The exception is the LP technique, with a relative error above 10% for almost all scenarios under consideration. In this regard, the comparison between the remaining time-based and count-based techniques also does not show significant variation, indicating that the lower performance of LP is related to its adaptive selection scheme. The mean throughput analysis of different sampling



(a) SystC 1/8



(b) SystC 1/64



(c) SystC 1/256

Figure 7.6: *Dispersion of estimated throughput of high workload - SystC*

techniques is particularly useful to guide measurement strategies for activities such as traffic engineering, accounting and SLA compliance, as measurements over large time intervals are considered.

Analyzing the packet size distribution, Table 7.8 shows that all techniques achieve accurate estimations of mean packet size, where the low workload scenario presents the less accurate results. The ratio between peak and average packet size, a descriptive statistic to measure the variability of packet time series, for identifying burstiness, also ratifies the accuracy of all techniques estimation. In this case, the low workload scenario exhibits the highest variability, while OC192 exhibits the less variable traffic along the measurement process, features correctly identified by the sampling techniques. For both metrics, count-based techniques (SystC and RandC) have presented more accurate results, followed by MuST, which improved the results obtained by time-based techniques.

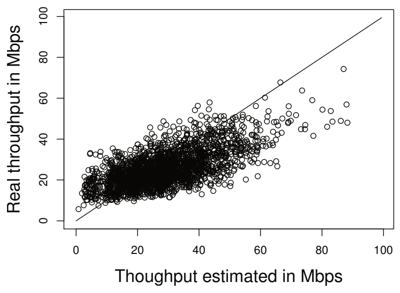
Despite the overall better results achieved by count-based techniques when estimating the mean throughput, the distribution of the instantaneous throughput estimation does not present the same tendency. In fact, adaptive techniques (*i.e.*, LP and MuST) are more stable for all traffic scenarios, providing high accuracy for short-time mea-

Table 7.8: *Overall traffic behavior - all sampling techniques*

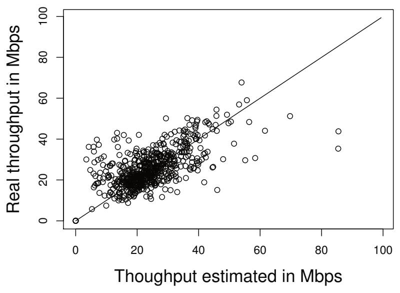
Parameter/Scenario	**Total**	**SystC**	**SystT**	**RandC**	**LP**	**MuST**
High workload						
Mean throughput (Mbps)	65.61	65.51	63.07	65.07	58.18	64.73
RME	-	0.0016	0.0388	0.0082	0.1132	0.0134
Mean packet size (Bytes)	648.59	647.95	633.33	643.15	635.49	652.36
Peak-to-average	2.33	2.33	2.39	2.35	2.38	2.32
Moderate workload						
Mean throughput (Mbps)	25.42	25.17	24.65	25.40	21.97	26.28
RME	-	0.0097	0.0302	0.0008	0.1355	0.0338
Mean packet size (Bytes)	587.26	586.90	579.42	586.53	589.38	587.82
Peak-to-average	2.57	2.57	2.61	2.58	2.56	2.57
Low workload						
Mean throughput (Mbps)	3.68	3.77	3.51	3.62	3.47	3.71
RME	-	0.0253	0.0460	0.0155	0.056	0.0091
Mean packet size (Bytes)	377.58	387.87	375.65	371.39	390.32	386.70
Peak-to-average	4.00	3.90	4.03	4.07	3.87	3.91
OC48						
Mean throughput (Mbps)	975.97	979.38	978.43	979.92	1085.12	985.29
RME	-	0.0035	0.0025	0.004	0.1118	0.0095
Mean packet size (Bytes)	565.44	567.31	566.84	567.94	561.71	566.18
Peak-to-average	3.67	3.58	3.66	3.65	3.69	3.66
OC192						
Mean throughput (Mbps)	1534.78	1535.57	1534.52	1537.07	1308.13	1515.29
RME	-	0.0005	0.0001	0.0014	0.1476	0.0126
Mean packet size (Bytes)	626.50	627.14	626.91	627.42	623.98	617.54
Peak-to-average	1.89	1.89	1.89	1.90	1.85	1.90

surement intervals. This analysis is illustrated in Figure 7.7 (for the moderate workload trace), corroborating the measurement improvement promoted by adaptive algorithms that rule the sampling selection considering the network activity variation. In this case, comparing both adaptive techniques, MuST (Figure 7.7(e)), outperforms LP (Figure 7.7(d)) providing an overall better result.

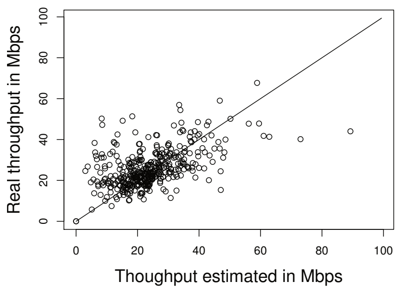
The outcome from the workload analysis has demonstrated that long-term and short-term measurements are better supported by different sampling techniques. While count-based techniques provide accurate estimations over long periods of network observation, the evaluated adaptive techniques are more indicated for measurements supporting near real-time activities.



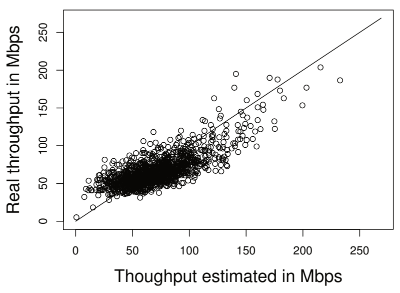
(a) SystC



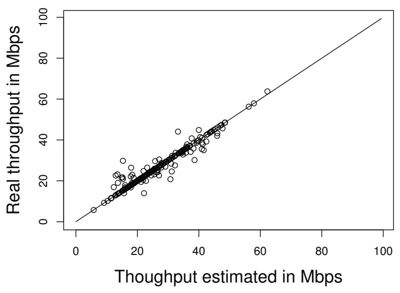
(b) SystT



(c) RandC



(d) LP



(e) MuST

Figure 7.7: *Dispersion of estimated throughput - Moderate workload*

7.5 Analyzing traffic flows through sampling

Apart from the workload analysis, the importance of traffic characterization for planning and managing effectively today's networks is undeniable. Associating network traffic with the corresponding applications and studying flows' characteristics allows gathering valuable information about network usage and, hence, devising solutions able to accommodate applications' requirements. In this context, the main objective of this section is to assess the applicability and performance of sampling techniques for traffic flow analysis. This involves analyzing the accuracy of statistics resulting from sampling techniques in capturing the characteristics of traffic flows crossing a network.

7.5.1 Methodology of tests

In order to assess the traffic sampling impact on flow analysis, the methodology of tests consists in applying different sampling techniques to real traffic scenarios, evaluating the estimation accuracy of common flow parameters. Similarly to previous sections, this study also extends the analysis to flow classification for different sampling frequencies of systematic techniques. The following subsections detail the traffic scenarios and the flow parameters used in the comparative study.

Traffic scenarios

The data used to evaluate sampling accuracy consists of four traffic traces collected from experimental distributed measurement system deployed in the INE network (presented in Section 6.3), in which multiple services are provided, such as videoconference, VoIP, distributed databases, private cloud, ftp, etc.

Each trace corresponds to a twenty minutes capture in different workload periods illustrating scenarios in which flow analysis is commonly used. The traffic traces are then handled as an aggregate, reflecting a continuous and heterogeneous network activity period of 252,087 individual unidirectional flows, comprising nearly 3 million packets.

Comparative parameters

For comparing the ability of distinct sampling techniques in assisting network flow analysis correctly, several flow parameters are considered, namely: (i) the amount of flows identified; (ii) the percentage of heavy-hitter (HH) flows identified, where the notion of heavy hitter refers to 20% of the largest flows in terms of size (number of packets) [106]; (iii) the utilization share at transport level; (iv) the utilization share at application level; and, (v) the accuracy of load estimations for the identified flows.

Considering that when flow characterization is based on sampling only a subset of the packets is available, estimating the underlying metrics involves the usage of statistical estimators to overcome missing data. In particular, the load estimation of each flow is an additional challenge as it needs to be often inferred from a small number of collected packets. Following the discussion in [71] and the notation in Table 7.9, the specific estimators in this comparative work are as follows:

- *Flow Mean Packet Size ($\bar{X}_f$):* the average of sampled packet sizes from flow f.

$$\bar{X}_f = \frac{\sum_{i=1}^{n_f} X_i}{n_f} \tag{7.3}$$

- *Estimated Flow Size (S_f):* the estimated number of packets in flow f.

$$S_f = N * \frac{n_f}{n_s} \tag{7.4}$$

- *Estimated Flow Load (L_f):* the estimated byte count of flow f.

$$L_f = S_f * \bar{X}_f \tag{7.5}$$

Table 7.9: *Notation*

X_i	the size of the ith sampled packet of flow f
n_f	number of sampled packets of flow f
n_s	total number of sampled packets
N	estimated total number of packets ($n_s/sampling\_frequency$)

Regarding the estimated flow load, this work applies an innovative way to assess accuracy by resorting to a nonparametric method to estimate the density distribution of load estimation (i.e., KDE - Kernel Density Estimation method) and thereby fostering the discussion on the estimation bias when applying each sampling technique. Each distribution corresponds to a nonparametric probability density function estimated using the Kernel method and a Gaussian smoothing scale. This method consists in drawing a continuous and smooth density distribution, weighted by the distance from a central value (the Kernel), where the population is inferred from a finite number of observations. In this context, as defined in [107], let $(L_{f1}, ..., L_{fn})$ be the estimated load of all identified flows (n) for which the density p is under evaluation. The shape of this function using the kernel estimator is given by:

$$\hat{p}_{bw}(L_f) = \frac{1}{n}\sum_{i=1}^{n} K(\frac{L_f - L_{fi}}{bw}), \tag{7.6}$$

where $K()$ is the kernel scaled by a Gaussian function, and bw is a smoothing parameter called bandwidth which defines the variance of the kernel in order to concentrate the density distribution within a specific interval. This interval is defined using the standard deviation of the smooth kernel when considering both unsampled traffic and traffic resulting from all sampling techniques.

When useful, the present study includes the Mean Square Error (MSE) of the estimated values, which is commonly used to evaluate the accuracy of estimators [71]. Note that, the evaluation of flow classification methodologies and tools is beyond the scope of this work, which resorts to a port-based classification technique for distinguishing flows. A work in progress within the research group is assessing the performance of sampling techniques in traffic classification based on DPI.

7.5.2 Identifying existing flows and heavy hitters

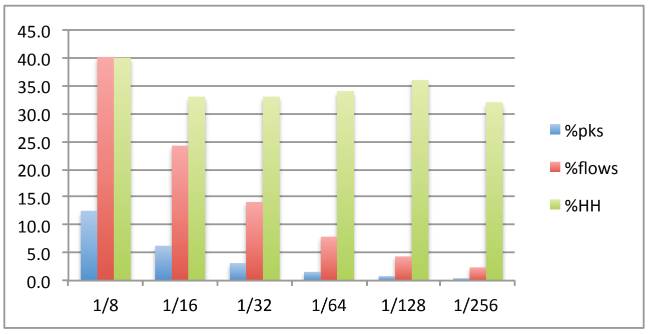
Intuitively, the sampling frequency is directly proportional to the estimation accuracy, as increasing the collected traffic results in a larger dataset for statistical analysis. While this may be true for analysis carried out using the same sampling technique, when the sampling selection changes, the trade-off between overhead and accuracy may differ significantly. Even within the same technique, studying this trade-off may bring useful information for reducing the amount of traffic collected and processed.

Considering the evaluation of SystC at different frequencies, the results in Table 7.10 confirm that decreasing the number of sampled packets leads to a decrease in the number of flows identified. As illustrated in Figure 7.8(a), while SystC 1/8 identifies 40% of existing flows, SystC 1/256 only detects 2,3% of flows. However, reducing the sampling frequency does not impact significantly on the accuracy in identifying the heavy-hitter flows. For instance, despite SystC 1/256 manipulates only 6% of the traffic processed by the technique SystC 1/16, the accuracy of heavy hitters identification is almost equivalent. This performance analysis is useful to guide activities in which accounting for heavy flows is relevant.

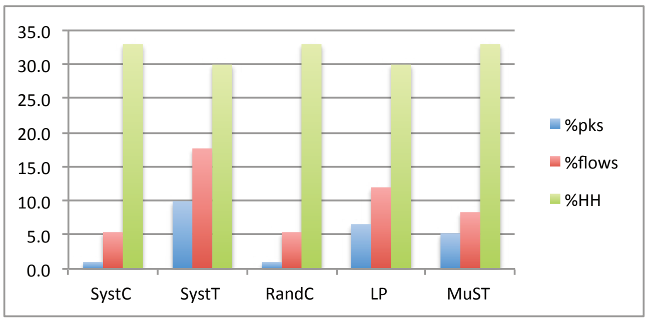
Table 7.10: *Identifying flows - comparative analysis*

	# Sampled packets	**# Distinct flows**	**% Heavy hitters**
SystC 1/8	373124	101168	40%
SystC 1/16	186562	61022	33%
SystC 1/32	93280	35523	33%
SystC 1/64	46639	19854	34%
SystC 1/128	23319	10891	36%
SystC 1/256	11665	5889	32%
SystC	29849	13652	33%
SystT	296579	44590	30%
RandC	29849	13577	33%
LP	196007	30186	30%
MuST	156382	21009	33%

Regarding the different sampling techniques, a larger number of sampled packets also implies a larger number of flows identified, as illustrated in Table 7.10. Nevertheless, count-based techniques are more efficient as, for the same proportion of sampled packets, the percentage of identified flows is significantly higher. This is visible when comparing the results of SystC 1/8 and SystT in Figures 7.8(a) and (b), respectively. This is due to the distinct packet selection policies in use, as the process of packet selection in count-based techniques increases the probability of capturing distinct flows, contrarily to time-based techniques in which packets are selected sequentially.



(a) SystC



(b) Sampling Techniques

Figure 7.8: *Identifying flows - comparative analysis*

Complementing the analysis of the accuracy in the number of flows identified, Figure 7.9 provides highlights regarding the flows' duration. The box plot graphic describes the distribution of the number of active flows observed in each sampling process in intervals of 1 second. As shown, time-based techniques maintain a more accurate view of existing flows along the time, although the large number of unsampled flows still affects significantly the instantaneous flow detection.

7.5.3 Utilization share at transport and application level

Even considering the reduced number of identified flows, the applicability of sampling in flow analysis may be useful for providing a realistic snapshot of the mix of protocols and applications in the network. To verify this argument, the sampling ac-

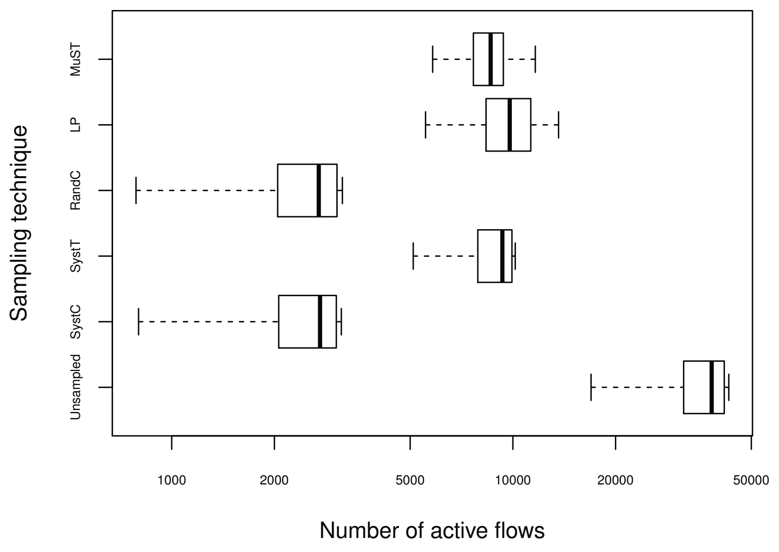


Figure 7.9: *Statistics on the number of active flows*

curacy analysis was extended to the context of traffic classification (both at transport and application level).

As presented in Figure 7.10 and 7.11, all sampling techniques and frequencies provide fairly accurate estimations of the most significant transport protocols in use (TCP, UDP). Regarding the analysis at transport level, the reduction on the number of sampled packets promoted by a lower sampling frequency of SystC does not affect accuracy, as presented in Figure 7.10. However, considering the application level, the estimated distribution is significantly affected, resulting in an overestimation of *http* traffic.

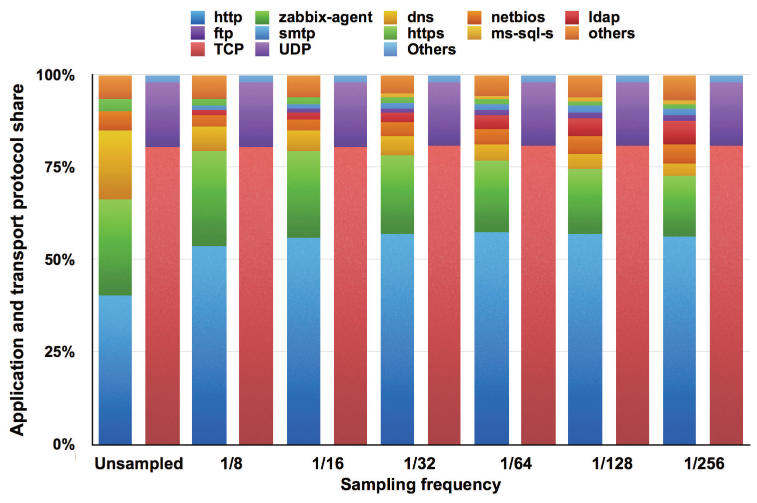


Figure 7.10: *Analysis at application and transport level - SystC*

Although changing the sampling technique also maintains the accuracy in classifying the transport protocol, the classification of applications presents more variability for the different techniques. As shown in Figure 7.11 and quantified in Table 7.11, in which the different techniques accuracy is assessed through MSE, time-based techniques lead to a more realistic distribution of the application share, with LP and MuST providing a slightly more accurate result. Globally, the results evince that an adequate yet small fraction of network traffic is able to provide a useful panoramic view of the protocolar mix of network flows.

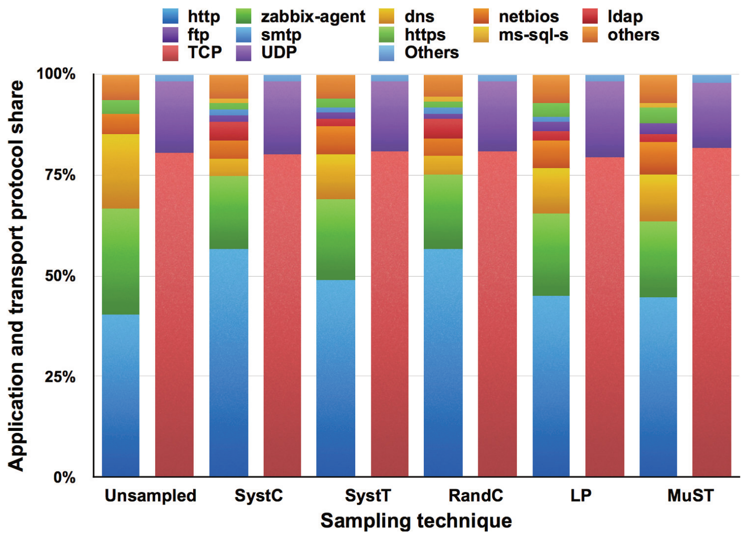


Figure 7.11: *Analysis at application and transport level - comparative techniques*

Table 7.11: *MSE - Traffic Classification*

	SystC	**SystT**	**RandC**	**LP**	**MuST**
Transport level	0.68	0.23	0.68	0.18	0.29
Application level	0.32	0.15	0.31	0.10	0.07

7.5.4 Load estimation

Attending to the formulation in Section 7.5.1, the results in Figure 7.12 show the distribution of the estimated flow load L_f (in logarithmic scale) when applying the different sampling techniques. The resulting graphics demonstrate the ability to represent the load distribution of all flows identified in the traffic trace, instead of only the more significant ones. This analysis plays a key role for traffic characterization and resource management activities.

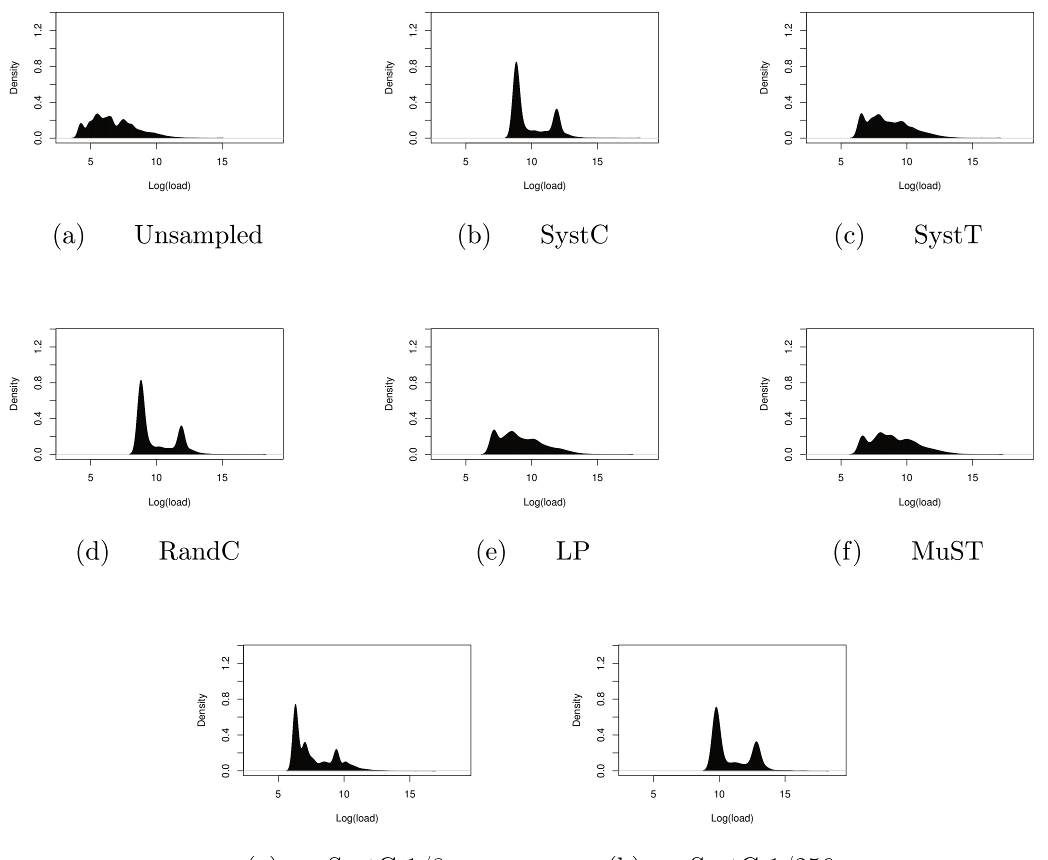
The results show that time-based techniques achieve a distribution closer to the real flow behavior (unsampled case in Figure 7.12 (a)) when compared with the count-based approaches (Figures 7.12 (b) and (d)), due to their more accurate load estimations of individual flows. This is observed through the better adjustment on the x-axis, meaning that the load estimations are closer to the real values. This suggests that a positive aspect (sparse packet selection) in flow identification becomes a drawback of the count-based techniques in flow dimensioning, since the current heuristics for flow load estimation consist in linear extrapolation proportional to the sampling frequency. This implies that a lower sampling frequency leads to overestimation of the flow load, deforming the density distribution to the right, and also concentrating the trend of load estimations (as detailed in Figure 7.12 (g) and (h), for two sampling frequencies of SystC technique), which evinces statistical loss of accuracy. This may interfere with network tasks in which the classification of small flows are of particular interest, such as intrusion detection and DDoS attacks.

Conversely, once time-based techniques select successive packets, the bursty behavior of flows tends to be better identified and dimensioned (when occurs into the sample size interval), resulting in more accurate flow load distributions, as presented in Figures 7.12(c), (e) and (f).

Table 7.12 includes statistics (mean, standard deviation and mode) to complement the above flow load analysis for the sampling techniques under study. As shown, the average flow load, the dispersion and the mode of the load estimations corroborate the behavior depicted in Figure 7.12. For the systematic count-based technique, as the sampling frequency decreases, the overestimation and concentration tendency is stressed.

Table 7.12: *Flow load statistical description*

	Mean	**Standard deviation**	**Mode**
Unsampled	6.08	1.48	4.18
SystC 1/8	7.26	1.40	6.26
SystC 1/16	7.91	1.39	6.89
SystC 1/32	8.61	1.38	7.62
SystC 1/64	9.34	1.37	8.33
SystC 1/128	10.06	1.37	9.03
SystC 1/256	10.79	1.38	9.73
SystC	9.78	1.36	8.79
SystT	8.04	1.47	4.49
RandC	9.78	1.37	8.79
LP	8.40	1.50	6.82
MuST	8.74	1.54	6.58



(a) Unsampled (b) SystC (c) SystT

(d) RandC (e) LP (f) MuST

(g) SystC 1/8 (h) SystC 1/256

Figure 7.12: *Density of flow load estimation*

7.6 Concluding remarks

Although the present evaluation study considers a limited set of possible measurement activities, the obtained results bring a valuable comparative insight among existing sampling techniques. The present contribution is therefore a step forward in providing a better understanding of traffic sampling techniques suitability and overhead regarding their deployment in real scenarios.

The results showed that despite the extensive deployment of systematic and random count-based techniques in current measurement tools, the adaptive and systematic time-based techniques can outperform them in some important aspects, such as the relationship between the volume of sampled data and computational resource usage (*i.e.*, CPU load and memory consumption), as well as flow identification and dimensioning.

The lack of a conclusive best sampling technique (in terms of overall performance), even considering a small set of measurement goals, ratifies the central purpose of this work. The performed analysis also demonstrated that, for some measurement goals, its is possible to reduce the amount of data involved in the network measurements without compromising the estimation accuracy. This confirms the framework versatility and potential in fostering the tuning and deployment of network measurement systems, revealing that a modular and configurable approach to sampling is a step forward for improving sampling scope and efficiency.

7.7 Summary

Demonstrating the versatility of the proposed sampling taxonomy in providing a flexible solution able to accommodate diverse sampling strategies, this chapter has presented a set of tests addressing the comparative performance of the main sampling techniques currently used in network measurement tools, *i.e.*, *systematic count-based* and *uniform random count-based*. In addition, other standard technique scarcely deployed *i.e.*, *systematic time-based* and two adaptive techniques were also evaluated, *i.e.*, *adaptive linear prediction* and *multiadaptive sampling*. The techniques were comparatively assessed regarding their computational weight (*i.e.*, CPU load, memory consumption and volume of data involved in the sampling process) and their accuracy in two activities usually supported by sampled measurements, *i.e.*, traffic workload and flow analysis. The overall results have shown that different sampling approaches achieve heterogeneous performance in distinct measurement activities, which ratifies the proposal of a modular and flexible sampling framework sustained by an encompassing taxonomy of sampling techniques. In addition, the chapter has also demonstrated the effectiveness of the proposed sampling technique (*i.e.*, MuST), outperforming conventional techniques in important aspects, such as the lower requirement of CPU and memory, as well as the higher accuracy in estimating flow dimension distribution.

Chapter 8

Conclusions

This chapter provides a concluding review of the present research work. At first, an assessment is carried out regarding to what extent the objectives outlined initially have been fulfilled, then the main contributions of the thesis are presented and debated. Finally, possible future research directions are pointed out.

8.1 Summary

The paradigm of having everyone and everything connected in an ubiquitous way poses huge challenges to today's networks due to the massive traffic volume involved and heterogeneity of the services provided. To turn treatable all network tasks requiring traffic analysis, traffic sampling has become a mandatory solution as only a subset of the packets traversing network devices is elected for analysis. A survey provided in Chapter 2 has demonstrated a large number of sampling techniques being proposed in last years aiming at achieving high-accuracy measurements in several network management tasks. However, despite the substantial research work in the area, the literature shows that there is not a single technique able to satisfy efficiently all measurement goals for diverse network scenarios, nor a flexible way to adapt sampling to specific measurement needs. This context has motivated the present research work, being its main objective defined as

> *devising an encompassing, flexible and lightweight traffic sampling architecture aiming at fostering the design and deployment of efficient sampling strategies for diverse traffic scenarios and measurement goals*

This has led to the design of a new multilayer sampling architecture based on a proposed sampling taxonomy able to sustain the modular deployment of sampling techniques. In the classification criterion, a set of features related to *sampling granularity*, *selection scheme* and *selection trigger* were identified and proposed as the main components distinguishing current and possibly forthcoming techniques.

Resorting to an experimental prototype, a set of proof-of-concept scenarios has demonstrated the potential of the proposed architecture in fostering the research and design of efficient sampling strategies.

8.2 Reviewing objectives and results

Considering the main objectives of this work and the identification of the relevant interrelated areas involved in conceiving an encompassing, flexible and lightweight sampling architecture, a set of objectives was formulated in Chapter 1. These objectives are here revisited and the main results obtained from their fulfillment are highlighted.

(i) To survey methods and tools for network and communication services measurement supported by packet sampling

This objective concerns to the bibliographic review surveying current main sampling techniques, the measurement tasks to which are oriented, its underlying metrics and performance achieved in those tasks. Having provided a consistent terminology of the main concepts related to packet sampling, the survey carried out in Chapter 2 has presented firstly the main standardized packet sampling schemes defined in RFC 5475 [1]. Then, the new proposals resulting from modifications and/or composition of basic approaches or entirely oriented to specific measurement goals were identified and classified. Considering specific tasks, these new proposals frequently outperform the main strategies (*i.e.*, *systematic count-based* and *random count-based*) currently available in research and commercial measurement tools. The large coverage of the related literature has also shown the lack of an encompassing study regarding the suitability of the available sampling techniques regarding multiple measurement tasks. This is mainly due to the complexity in deploying the vast number of techniques in current measurement tools. The analysis carried out under this objective has grounded the motivation for the present research study, pointing out the main strategic directions to follow toward a new multilayer sampling architecture. In addition, fostered by the literature review, a new Multiadaptive sampling technique was proposed, having as its main characteristic, the ability to self-adjust the sampling policy and thus reducing the computational overhead without compromising the measurement accuracy.

(ii) To propose a comprehensive model for flexible deployment of current and forthcoming sampling techniques

After identifying that most of the sampling techniques, whether simple or complex, share a set of structural components, these components are described in Chapter 4 through a hierarchical taxonomy. The defined taxonomy fragments the sampling techniques 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 from which both classic and recently proposed techniques can be described by the combination of, at least, one approach of each component. This structure sustains the modular and flexible deployment of a comprehensive number of techniques in order to exploit specific features aiming at improving the balance between accuracy and computational overhead in different measurement and traffic scenarios. The chapter ends presenting a classification summary of the main surveyed techniques according to the proposed taxonomy, confirming its suitability in providing a comprehensive and flexible model for sampling techniques deployment.

(iii) To conceive and specify a sampling-based architecture able to accommodate the proposed taxonomy and the several systems involved in network and communication services measurements

The achievement of this objective, addressed in Chapter 5, resulted in the definition and specification of an encompassing measurement architecture able to suit the following design goals: (*i*) compatibility with current protocols and measurement tools in order to support its deployment in current measurement systems; (*ii*) specification and deployment sustained by open and standard protocols; (*iii*) flexibility to support the introduction of new protocols and technologies related to traffic sampling; (*iv*) versatility to deploy modularly current and forthcoming sampling techniques; and (*v*) capability to support mechanisms for balancing measurement accuracy and computational weight in order to foster the design of efficient sampling strategies. The architecture is composed by three layers (*i.e.*, *management plane*, *control plane* and *data plane*) covering all the components involved in traffic measurements and their underlying interactions. In addition, the architecture specification also covered aspects related to mapping the measurement needs into the most suitable sampling technique. Aspects such as the sampling frequency, the decision on the packet fields of interest, the level of aggregation and the exporting format according to the network scenario and measurement goals to fulfill are also discussed.

(iv) To implement a prototype of the proposed architecture

This objective is a natural step that precedes the validation of the sampling architecture. The use of a prototype allows a comprehensive evaluation of the ability in designing sampling strategies through a modular and flexible structure. Accomplishing this objective, addressed in Chapter 6, led to the identification and discussion of the main implementation aspects concerning the computational architecture and the development of a functional sampling framework based on the proposed taxonomy. This framework provides a fair environment in which several sampling techniques can be comparatively assessed aiming at designing efficient measurement strategies for diverse traffic scenarios. The discussion ends presenting an experimental distributed measurement system running in a large-scale network and a complete demonstration of performing network traffic accounting through the proposed architecture.

(v) To provide a proof-of-concept environment aiming at evaluating the effectiveness of the proposed architecture

The proof-of-concept of the proposed sampling architecture had as main target to demonstrate the versatility of the proposed sampling taxonomy in providing a flexible solution able to accommodate diverse sampling strategies. A second objective was to elaborate a fair comparative evaluation of different sampling techniques when applied to activities usually supported by sampling. The performance was assessed considering the accuracy in metric estimation of the related activities (*i.e.*, traffic workload and flow analysis) and the computational requirements (*i.e.*, CPU load, memory consumption and volume of data involved) in performing such techniques. The corresponding results, covered in Chapter 7, were obtained resorting to real traffic scenarios and reported the performance of different sampling approaches when applied to distinct measurement tasks. Considering that some of the less deployed techniques in current measurement tools can improve significantly the measurement efficiency in various network conditions, the results have also demonstrated the framework versatility and potential in fostering the deployment and tuning of network measurement systems, which ratifies the central purpose of this work.

8.3 Main contributions

Taking into account the initial objectives and obtained results, the main contributions of this work are summarized below.

- *Proposal of a new multiadaptive sampling technique* - A new multiadaptive technique able to self-adjust the packet sampling policy according to the observed traffic activity has been devised [14] [15] [16] [17]. Its main innovation consists in the ability to reduce resource requirements in periods of higher network activities without compromising estimation accuracy. The technique is based on linear prediction of future network activity and its multiadaptive property is achieved considering both the *interval between samples* and the *sample size* as adaptive parameters. Both adjustable parameters are bounded by appropriate thresholds in order to guarantee the representativeness of samples in capturing the network behavior. The outcome of the proof-of-concept has evinced the effectiveness of the proposal, outperforming the conventional techniques, mainly in saving resources (*i.e.*, CPU and memory) in high workload scenarios. The technique has also demonstrated high accuracy in estimating most of the metrics under analysis, with significant improvement in dimensioning individual flows from the aggregated traffic;

- *Proposal of a taxonomy of packet sampling techniques* - Although the classic sampling techniques were previously classified through RFC 5475 [1], the document does not cover recent advances in packet sampling introduced by new proposals (*e.g.*, adaptive techniques). In this way, the conceived taxonomy [18] [19] describes current sampling techniques through its constituent parts, rather than a closed unit, allowing to identify their common properties and address eventual constraints within a narrower and simpler scope. In the classification criterion, a set of features related to sampling *granularity*, *selection scheme* and *selection trigger* are identified and proposed as the main components distinguishing current proposals. Then, each component is further divided into a set of approaches. Following the taxonomy structure it is possible to drive a modular deployment of most sampling techniques through the combination of proper approaches. Beyond the comprehensiveness in classifying the current sampling techniques, this model allows the design of sampling strategies able to exploit specific features that lead to better performance for each measurement purpose and traffic scenario;

- *Proposal of an encompassing, flexible and lightweight packet sampling architecture* - Supported by the flexibility introduced by the proposed taxonomy, a sampling-based measurement architecture has been devised [18] [20]. The architecture is composed of three layers (*i.e.*, *management plane*, *control plane* and *data plane*) covering all the elements involved in traffic measurements. Each layer is modularly designed aiming at supporting the required compatibility with several measurement protocols and tools. The modular structure also provides flexibility to accommodate mechanisms able to enhance the overall performance in current and forthcoming measurement goals. In this way, all the architectural components were presented along with the different strategies and technologies that compose the several stages of a measurement process. Thus, the main contribution of the architecture specification relies on the completeness in addressing aspects regarding: (*i*) the identification of the most suitable sampling technique and measurement point toward a specific measurement goal and traffic scenario; (*ii*) the modular deployment of sampling techniques; (*iii*) the support of several related protocols and technologies; (*iv*) the lightweight packet capturing; (*v*) the strategies for aggregation and exporting sampled data; (*vi*) processing and estimation of measurement metrics; and (*vii*) strategies supporting the deployment of a distributed measurement system;

- *Development of a sampling framework* - A functional sampling framework able to deploy different techniques following the taxonomy structure has been implemented [21] [22]. This framework provides a fair environment in which different sampling techniques can be comparatively assessed in order to identify the most suitable for each measurement goal and traffic scenario. This is a fundamental aspect in order to support the design of efficient measurement strategies based on packet sampling. The developed framework also demonstrates the lightweight feature of the proposed architecture through its current deployment in low cost and low power devices (*i.e.*, Raspberry Pi), being used to performing large scale network monitoring (at INE network);

- *Provide a fair comparative analysis regarding the performance of different techniques in diverse traffic scenarios* - Supported by the developed framework, a set of comparative analyses of the computational weight and achieved accuracy in activities usually assisted by packet sampling has been carried out [20] [21] [22] [23]. These tests have provided valuable outcomes in order to guide the selection of the most suitable sampling technique for traffic workload and classification activities. The results, detailed in Chapter 7, have mainly demonstrated that despite the extensive deployment of systematic and random count-based techniques in current measurement tools, some recent techniques usually outperform them in several scenarios without additional overhead. For instance, the multiadaptive and systematic time-based techniques achieved a better relationship between the volume of sampled data and the computational resources (*i.e.*, CPU and memory) required. These techniques have also improved the estimation accuracy in flow identification and dimensioning.

8.4 Topics of future work

From the analysis carried out along this work, several topics have been identified as deserving further research. The following list comprises topics which may improve or extend particular aspects of the proposed solution.

- Having demonstrated the ability to implement several sampling techniques, future work intends to drive extensive and systematic comparative analysis toward each measurement task supported by packet sampling (identified in Chapter 2). The aim is to provide reliable inputs in order to select the most suitable technique for specific scenarios and measurement goals. Considering that most measurement tools are not tailored for sampled data inputs, it will also require efforts in tuning the methods for the underlying metrics' estimation;

- Even understanding the expected performance of each sampling technique when applied to various measurement goals and traffic scenarios, deciding for a specific strategy is still a challenging task. As discussed along with Chapter 5, this decision also involves: (*i*) the sampling techniques available at each MP; (*ii*) the resource requirements the technique imposes facing the MP device architecture; (*iii*) conflicting measurement needs from different network tasks performing sampling in the same MP; and (*iv*) definition of the MPs that will participate in the sampling process. In this way, future work will consider developing an automatic system able to balance all these aspects in order to drive the mapping of the measurement needs into the sampling technique and measurement point. This mechanism will be sustained by formal definition of the measurement and sampling entities (and their attributes) resorting to an ontology currently under development. The usage of ontologies will provide flexibility in balancing the dynamic nature of concepts such as measurement needs, estimated accuracy, overhead and performance.

- Due to the difficulty in deploying innovative components introduced by the defined architecture in current measurement tools, future work intends to explore SDN features in order to endow these systems with the proposed capabilities. The main idea consists in taking advantage of the SDN device programability to implement the modular configuration of sampling techniques defined along this work. To pursue this, a set of counters (*e.g.*, temporal counters and packet counters) and actions (*e.g.*, explicit packet dispatch to the controller) specified in OpenFlow [108] will be systematically arranged toward packet sampling. This strategy does not require any modification in current SDN devices, which is in accordance with the design goal of keeping the data plane simple, avoiding processing overhead. The remaining modules defined in the architecture can thus be deployed in general or dedicate measurement SDN controllers, depending on the measurement goals and available resources.

- Despite the efforts in taking the proposed sampling architecture to SDN enable devices, it is also objective of future work to evolve the framework prototype to a complete standalone tool. Following the overall design goals and architecture specification, it is expected to obtain a tool easy to be deployed in general purpose devices, improving the effectiveness of the network measurements with low associated costs. This will also involve the development of an API in order to sustain the easy composition of new sampling techniques and a plug-in based system to support the addition of new architectural modules.

8.5 Final considerations

The present research work was mainly focused on fostering the efficiency of network measurements through the development of a modular architecture for flexible deployment of packet sampling strategies. Although there is a lack of a study identifying the best sampling approach for each measurement goal and traffic scenario, the coverage of the literature and the outcomes from this work have clearly demonstrated the need for a manifold solution. In this way, the resultant architecture, sustained by a modular taxonomy of sampling techniques, provides a valuable contribution to the research field. The topics pointed out as future work, aiming at promoting the solution to current and forthcoming measurement environments, reveal interesting research directions remaining ahead.

Appendix A

MuST Algorithm

This appendix includes the pseudocode for Algorithm 1, which allows to predict the reference parameter, and for Algorithm 2, which represents the overall sampler operation. This algorithm allows to determine the new sample size and interval between samples, attending to the predicted reference parameter.

```
input  : $X$ - Reference parameter vector
         $\Delta T$ - Current time interval between samples
         $T$ - Times vector
output: forecast - Reference parameter prediction
1 for $i \leftarrow 1$ to $order - 1$ do
2     $sum \leftarrow sum + abs((X[i+1] - X[i])/T[i])$;
3 end
4 $forecast \leftarrow X[order] + ((\Delta T/(order - 1)) * sum$;
5 return ($forecast$);
```

Algorithm 1: Pseudocode for the reference parameter predictor

```
input  : $\Delta T_{current}$ - Initial interval between samples
         $\Delta S_{current}$ - Initial sample size
output: Sampled traffic
begin
    for $i \leftarrow 1$ to order do
        newSample($\Delta T_{current}$, $\Delta S_{current}$); /* new sample capture */;
        $X[i] \leftarrow$ referenceParameter(); /* stores ref. par. of last sample */;
        $T[i] \leftarrow \Delta T_{current}$;  /* kept unchanged */
    end
    repeat
        /* Predictor() corresponds to Algorithm 1 */;
        $X_p \leftarrow$ Predictor($X$, $\Delta T$, $T$);
        newSample ($\Delta T_{current}, \Delta S_{current}$);
        $S \leftarrow$ referenceParameter();
        $m \leftarrow 1$;
        if $S$ then
            $m \leftarrow X_p/S$
        end
        case [$m < m_{min}$]  /* underestimation */
          $\Delta T_{next} \leftarrow m * \Delta T_{current}$;
          $\Delta S_{next} \leftarrow m * \Delta S_{current}$;
        case [$m_{min} \leq m \leq m_{max}$] /* correct estimation*/
          $\Delta T_{next} \leftarrow \Delta T_{current}$;
          $\Delta S_{next} \leftarrow \Delta S_{current}$;
        case [$m > m_{max}$]  /* overestimation */
          $\Delta T_{next} \leftarrow 2 * \Delta T_{current}$; $k$=0.15;
          $\Delta S_{next} \leftarrow (1 + k) * \Delta S_{current}$;
        /* Interval Between Samples thresholds (sec) */
        if $\Delta T_{next} < MinIBS$ then
            $\Delta T_{next} \leftarrow MinIBS$ ;
        end
        if $\Delta T_{next} > MaxIBS$ then
            $\Delta T_{next} \leftarrow MaxIBS$;
        end
        /* Sampling Size thresholds (sec) */
        if $\Delta S_{next} < MinSS$ then
            $\Delta S_{next} \leftarrow MinSS$;
        end
        if $\Delta S_{next} > MaxSS$ then
            $\Delta S_{next} \leftarrow MaxSS$;
        end
        $\Delta T_{current} \leftarrow \Delta T_{next}$; $\Delta S_{current} \leftarrow \Delta S_{next}$;
        updateVectors($\Delta T_{current}, S$); /* update vectors T and X */
    until endOfSampling;
end
```

Algorithm 2: Multiadaptive technique pseudocode

Bibliography

[1] T. Zseby, M. Molina, and N. Duffield, "Sampling and Filtering Techniques for IP Packet Selection RFC 5475," IETF, Tech. Rep., 2009. [Online]. Available: http://datatracker.ietf.org/doc/rfc5475/

[2] N. Duffield, "Sampling for Passive Internet Measurement: A Review," *Statistical Science*, vol. 19, no. 3, pp. 472–498, Aug. 2004. [Online]. Available: http://projecteuclid.org/Dienst/getRecord?id=euclid.ss/1110999311/

[3] Z. Qian, H. Li, R. Tang, and K. W. Cheung, "Performance of traffic aggregation versus segregation for Optical Flow Switching networks," in *2011 International Conference on Information Photonics and Optical Communications*. IEEE, Oct. 2011, pp. 1–3. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6122860

[4] M. Grossglauser and J. Rexford, "Passive Traffic Measurement for IP Operations," in *The Internet as Large-Scale Complex System*. Oxford University Press, 2002, p. 2.

[5] S. K. Thompson, *Sampling*, 2nd ed. Wiley-Interscience, 2002.

[6] D. Tammaro, S. Valenti, D. Rossi, and A. Pescapé, "Exploiting packet-sampling measurements for traffic characterization and classification," *International Journal of Network Management*, vol. 22, no. 6, pp. 451–476, Nov. 2012. [Online]. Available: http://dx.doi.org/10.1002/nem.1802

[7] T. Zseby, T. Hirsch, and B. Claise, "Packet Sampling for Flow Accounting: Challenges and Limitations," in *Passive and Active Network Measurement*, ser. Lecture Notes in Computer Science, M. Claypool and S. Uhlig, Eds. Springer Berlin / Heidelberg, 2008, vol. 4979, pp. 61–71. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-79232-1_7

[8] C. Hu, S. Wang, J. Tian, B. Liu, Y. Cheng, and Y. Chen, "Accurate and Efficient Traffic Monitoring Using Adaptive Non-Linear Sampling Method," in *IEEE INFOCOM 2008 - The 27th Conference on Computer Communications*. IEEE, Apr. 2008, pp. 26–30. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4509609

[9] A. N. Mahmood, J. Hu, Z. Tari, and C. Leckie, "Critical infrastructure protection: Resource efficient sampling to improve detection of less frequent patterns in

network traffic," *Journal of Network and Computer Applications*, vol. 33, no. 4, pp. 491–502, 2010. [Online]. Available: http://www.sciencedirect.com/science/article/B6WKB-4YBMFB6-1/2/9b91d8daa2364e0d025aed60881 60da7

[10] J. Sommers, P. Barford, N. Duffield, and A. Ron, "Improving accuracy in end-to-end packet loss measurement," in *Proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for computer communications - SIGCOMM '05*, vol. 35, no. 4. New York, New York, USA: ACM Press, Aug. 2005, p. 157. [Online]. Available: http://dl.acm.org/citation.cfm?id=1080091.1080111

[11] Z.-G. Hu, D.-L. Zhang, C.-P. Hou, and J.-S. Zhang, "Adaptive sampling algorithm of network round-trip time," in *Journal of Computer Applications*, J. Yingyong, Ed., vol. 30, no. 2, 2010, pp. 319–322.

[12] X. He, W. Yang, and Q. Wang, "An Adaptive Traffic Sampling Method for Anomaly Detection," in *2009 Fourth International Conference on Internet Computing for Science and Engineering*. IEEE, Dec. 2009, pp. 142–146. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5521614

[13] G. Androulidakis, V. Chatzigiannakis, and S. Papavassiliou, "Network anomaly detection and classification via opportunistic sampling," *IEEE Network*, vol. 23, no. 1, pp. 6–12, Jan. 2009. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4804318

[14] J. M. C. Silva, P. Carvalho, and S. Rito Lima, "A multiadaptive sampling technique for cost-effective network measurements," *Computer Networks*, vol. 57, no. 17, pp. 3357–3369, Dec. 2013. [Online]. Available: http://dx.doi.org/10.1016/j.comnet.2013.07.023http://linkinghub.elsevier.com/retrieve/pii/S1389128613002491

[15] J. M. C. Silva and S. R. Lima, "Multiadaptive Sampling for Lightweight Network Measurements," in *2012 21st International Conference on Computer Communications and Networks (ICCCN)*. IEEE, Jul. 2012, pp. 1–7. [Online]. Available: http://ieeexplore.ieee.org/articleDetails.jsp?arnumber=6289265

[16] J. M. C. Silva and S. R. Lima, "Improving Network Measurement Efficiency through Multiadaptive Sampling," in *Traffic Monitoring and Analysis SE - 18*, ser. Lecture Notes in Computer Science, A. Pescapè, L. Salgarelli, and X. Dimitropoulos, Eds. Springer Berlin Heidelberg, 2012, vol. 7189, pp. 171–174. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-28534-9_18

[17] ——, "Optimizing Network Measurements through Self-adaptive Sampling," in *2012 IEEE 14th International Conference on High Performance Computing and Communication*. IEEE, Jun. 2012, pp. 794–801. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6332250

[18] J. M. C. Silva, P. Carvalho, and S. Rito Lima, "Enhancing Traffic Sampling scope and efficiency," in *2013 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS).* IEEE, Apr. 2013, pp. 71–72. [Online]. Available: http://ieeexplore.ieee.org/articleDetails.jsp?arnumber=6562848

[19] ——, "Inside Packet Sampling Techniques: Exploring Modularity to Enhance Network Measurements," *International Journal of Communication Systems*, vol. In press, 2015.

[20] ——, "A modular architecture for deploying self-adaptive traffic sampling," in *8th IFIP WG 6.6 International Conference on Autonomous Infrastructure, Management, and Security, AIMS 2014.* Brno: Springer, 2014, pp. 179–183. [Online]. Available: 10.1007/978-3-662-43862-6_21

[21] J. M. C. Silva, P. Carvalho, and S. R. Lima, "Computational weight of network traffic sampling techniques," in *2014 IEEE Symposium on Computers and Communications (ISCC).* Madeira, Portugal: IEEE, Jun. 2014, pp. 1–6. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6912467

[22] J. M. C. Silva, P. Carvalho, and S. Rito Lima, "A Modular Sampling Framework for Flexible Traffic Analysis," in *The 23rd International Conference on Software, Telecommunications and Computer Networks - SoftCOM*, Split, 2015.

[23] ——, "Analysing Traffic Flows Through Sampling: A Comparative Study," in *20th IEEE Symposium on Computers and Communication (ISCC)*, Cyprus, 2015.

[24] J. Jadwab, P. Phall, and B. Pinna, "Traffic estimation for the largest sources on a network using packet sampling with limited storage," Hewllet-Packard Laboratories, Bristol, Tech. Rep., 1992. [Online]. Available: http://groshens.org/Whitepaper/architecture/HPL-92-35to$\delimiter"026E30F$manage$\delimiter"026E30F$networks.pdf

[25] K. C. Claffy, G. C. Polyzos, and H.-W. Braun, "Application of sampling methodologies to network traffic characterization," *SIGCOMM Comput. Commun. Rev.*, vol. 23, no. 4, pp. 194–203, 1993. [Online]. Available: http://doi.acm.org/10.1145/167954.166256

[26] I. Cozzani and S. Giordano, "Traffic sampling methods for end-to-end QoS evaluation in large heterogeneous networks," *Computer Networks and ISDN Systems*, vol. 30, no. 16-18, pp. 1697–1706, Sep. 1998. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0169755298001986

[27] P. Amer and L. Cassel, "Management of sampled real-time network measurements," in *[1989] Proceedings. 14th Conference on Local Computer Networks.* IEEE Comput. Soc. Press, 1989, pp. 62–68. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=65244

[28] E. A. Hernandez, M. C. Chidester, and A. D. George, "Adaptive Sampling for Network Management," *Journal of Network and Systems Management*, vol. 9, no. 4, pp. 409–434, 2001. [Online]. Available: http://dx.doi.org/10.1023/A:1012980307500

[29] N. G. Duffield and M. Grossglauser, "Trajectory sampling for direct traffic observation," *ACM SIGCOMM Computer Communication Review*, vol. 30, no. 4, pp. 271–282, Oct. 2000. [Online]. Available: http://dl.acm.org/citation.cfm?id=347057.347555

[30] C. Estan and G. Varghese, "New directions in traffic measurement and accounting," *SIGCOMM Comput. Commun. Rev.*, vol. 32, no. 4, pp. 323–336, Oct. 2002. [Online]. Available: http://dl.acm.org/citation.cfm?id=964725.633056http://doi.acm.org/10.1145/964725.633056

[31] R. Singh, Raman and Kumar, Harish and Singla, "Analyzing Statistical Effect of Sampling on Network Traffic Dataset," in *ICT and Critical Infrastructure: Proceedings of the 48th Annual Convention of Computer Society of India*, S. Satapathy, Suresh Chandra and Avadhani, P. S. and Udgata, Siba K. and Lakshminarayana, Ed. Springer International Publishing, 2014, pp. 401–408. [Online]. Available: http://link.springer.com/chapter/10.1007/978-3-319-03107-1_43

[32] L. Yang and G. Michailidis, "Sampled Based Estimation of Network Traffic Flow Characteristics," in *IEEE INFOCOM 2007 - 26th IEEE International Conference on Computer Communications*. IEEE, 2007, pp. 1775–1783. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4215789

[33] V. Carela-Español, P. Barlet-Ros, A. Cabellos-Aparicio, and J. Solé-Pareta, "Analysis of the impact of sampling on NetFlow traffic classification," *Computer Networks*, vol. 55, no. 5, pp. 1083–1099, Apr. 2011. [Online]. Available: http://dx.doi.org/10.1016/j.comnet.2010.11.002

[34] R. Lin, O. Li, Q. Li, and K. Dai, "Exploiting Adaptive Packet-Sampling Measurements for Multimedia Traffic Classification," *Journal of Communications*, vol. 9, no. 12, 2014. [Online]. Available: http://www.jocm.us/uploadfile/2014/1231/20141231030404520.pdf

[35] S. Kandula and R. Mahajan, "Sampling biases in network path measurements and what to do about it," in *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*, ser. IMC '09. New York, NY, USA: ACM, 2009, pp. 156–169. [Online]. Available: http://doi.acm.org/10.1145/1644893.1644912

[36] M. Lee, N. Duffield, and R. Kompella, "Two Samples are Enough: Opportunistic Flow-level Latency Estimation using NetFlow," in *2010 Proceedings IEEE INFOCOM*. IEEE, Mar. 2010, pp. 1–9. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5462044

[37] J. Zhang, X. Luo, R. Perdisci, G. Gu, W. Lee, and N. Feamster, "Boosting the scalability of botnet detection using adaptive traffic sampling," in *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security*, ser. ASIACCS '11. New York, NY, USA: ACM, 2011, pp. 124–134. [Online]. Available: http://doi.acm.org/10.1145/1966913.1966930

[38] Yih Huang and J. Pullen, "Countering denial-of-service attacks using congestion triggered packet sampling and filtering," in *Proceedings Tenth International Conference on Computer Communications and Networks (Cat. No.01EX495)*, IEEE. IEEE, 2001, pp. 490–494. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=956309

[39] D. Brauckhoff, B. Tellenbach, A. Wagner, M. May, and A. Lakhina, "Impact of packet sampling on anomaly detection metrics," in *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, ser. IMC '06. New York, NY, USA: ACM, 2006, pp. 159–164. [Online]. Available: http://doi.acm.org/10.1145/1177080.1177101

[40] I. Paredes-Oliva, P. Barlet-Ros, and J. Solé-Pareta, "Portscan detection with sampled netflow," in *Traffic Monitoring and Analysis*. Springer, 2009, pp. 26–33. [Online]. Available: http://link.springer.com/chapter/10.1007/978-3-642-01645-5_4

[41] J. Mai, C.-N. Chuah, A. Sridharan, T. Ye, and H. Zang, "Is sampled data sufficient for anomaly detection?" in *Proceedings of the 6th ACM SIGCOMM on Internet measurement - IMC '06*, ser. IMC '06. New York, New York, USA: ACM Press, 2006, p. 165. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1177080.1177102

[42] J. Jae-Hyun, A. Cheol-Woong, L. Dongjoon, and K. Sung-Ho, "DDoS Attack Detection Using Flow Entropy and Packet Sampling on Huge Networks," in *ICN 2014 : The Thirteenth International Conference on Networks*. IARIA, 2014, pp. 183–190.

[43] T. Zseby, "Deployment of sampling methods for SLA validation with non-intrusive measurements," in *Proceedings of Passive and Active Measurements Conference*, Fort Collins, 2002.

[44] ——, "Comparison of sampling methods for non-intrusive SLA validation," in *Proceedings of the Second Workshop on End-to-End Monitoring Techniques and Services (E2EMon)*, 2004.

[45] R. Serral-Gracia, A. Cabellos-Aparicio, and J. Domingo-Pascual, "Packet Loss Estimation Using Distributed Adaptive Sampling," in *Network Operations and Management Symposium Workshops, 2008. NOMS Workshops 2008. IEEE*. IEEE, Apr. 2008, pp. 124–131. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4509938

[46] A. Dogman, R. Saatchi, and S. Al-Khayatt, “An adaptive statistical sampling technique for computer network traffic,” in *Communication Systems Networks and Digital Signal Processing (CSNDSP), 2010 7th International Symposium on*, 2010, pp. 479–483. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5580380

[47] Y. Gu, L. Breslau, N. Duffield, and S. Sen, “On Passive One-Way Loss Measurements Using Sampled Flow Statistics,” in *IEEE INFOCOM 2009 - The 28th Conference on Computer Communications*. IEEE, Apr. 2009, pp. 2946–2950. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5062264

[48] I. O. S. Cisco, “NetFlow,” 2008.

[49] C. Henke, C. Schmoll, and T. Zseby, “Empirical Evaluation of Hash Functions for PacketID Generation in Sampled Multipoint Measurements,” in *Passive and Active Network Measurement*, ser. Lecture Notes in Computer Science, S. Moon, R. Teixeira, and S. Uhlig, Eds. Springer Berlin / Heidelberg, 2009, vol. 5448, pp. 197–206. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-00975-4_20

[50] T. Zseby, “Stratification Strategies for Sampling-based Non-intrusive Measurement of One-way Delay,” in *Proceedings of Passive and Active Measurement Workshop (PAM 2003), La Jolla, CA, USA*. Citeseer, 2003, pp. 171–179.

[51] K. Ishibashi, R. Kawahara, M. Tatsuya, T. Kondoh, and S. Asano, “Effect of sampling rate and monitoring granularity on anomaly detectability,” in *2007 IEEE Global Internet Symposium*. IEEE, May 2007, pp. 25–30. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4301426

[52] E. Cox, M. O’Hagan, R. Taber, and M. O’Hagen, *The Fuzzy Systems Handkbook with Cdrom*. Academic Press, Inc., 1998.

[53] R. Saatchi and A. Dogman, “Multimedia traffic quality of service management using statistical and artificial intelligence techniques,” *IET Circuits, Devices & Systems*, vol. 8, no. 5, pp. 367–377, Sep. 2014. [Online]. Available: http://digital-library.theiet.org/content/journals/10.1049/iet-cds.2013.0454

[54] W. E. Leland, W. Willinger, M. S. Taqqu, and D. V. Wilson, “On the self-similar nature of Ethernet traffic,” *ACM SIGCOMM Computer Communication Review*, vol. 25, no. 1, pp. 202–213, Jan. 1995. [Online]. Available: http://dl.acm.org/citation.cfm?id=205447.205464

[55] A. Pescape, D. Rossi, D. Tammaro, and S. Valenti, “On the impact of sampling on traffic monitoring and analysis,” in *2010 22nd International Teletraffic Congress (lTC 22)*. IEEE, Sep. 2010, pp. 1–8. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5608718

[56] Y. Chabchoub, C. Fricker, F. Guillemin, and P. Robert, “Deterministic Versus Probabilistic Packet Sampling in the Internet,” in *Managing Traffic Performance in Converged Networks*, ser. Lecture Notes in Computer

Science, L. Mason, T. Drwiega, and J. Yan, Eds. Springer Berlin / Heidelberg, 2007, vol. 4516, pp. 678–689. [Online]. Available: http://link.springer.com/chapter/10.1007/978-3-540-72990-7_60

[57] S. Shirali-Shahreza and Y. Ganjali, “FleXam,” in *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking - HotSDN ’13*. New York, New York, USA: ACM Press, Aug. 2013, p. 167. [Online]. Available: http://dl.acm.org/citation.cfm?id=2491185.2491215

[58] N. Hohn and D. Veitch, “Inverting sampled traffic,” in *Proceedings of the 2003 ACM SIGCOMM conference on Internet measurement - IMC ’03*. New York, New York, USA: ACM Press, Oct. 2003, p. 222. [Online]. Available: http://dl.acm.org/citation.cfm?id=948205.948235

[59] P. Phaal, S. Panchen, and N. McKee, “InMon Corporation’s sFlow: A Method for Monitoring Traffic in Switched and Routed Networks,” RFC 3176 (Informational), Sep. 2001. [Online]. Available: http://www.ietf.org/rfc/rfc3176.txt

[60] A. Schulman, D. Levin, and N. Spring, “{CRAWDAD} trace umd/sigcomm2008/pcap/Ethernet (v. 2009-03-02),” Downloaded from http://crawdad.cs.dartmouth.edu/umd/sigcomm2008/pcap/Ethernet, Mar. 2009.

[61] C. Shannon, E. Aben, K. claffy, D. Andersen, and N. Brownlee, “The CAIDA UCSD Anonymized Internet Traces 2008 - equinix-chicago.dirA.20080430-170200.UTC.anon,” Downloaded from http://www.caida.org/data/passive/passive_2008_dataset.xml, Apr. 2008.

[62] J. Rajahalme, A. Conta, B. Carpenter, and S. Deering, “IPv6 Flow Label Specification,” RFC 3697, Internet Engineering Task Force, Tech. Rep. 3697, 2004. [Online]. Available: http://datatracker.ietf.org/doc/rfc3697/

[63] S. Handelman, S. Stibler, N. Brownlee, and G. Ruth, “RTFM: New Attributes for Traffic Flow Measurement,” RFC 2724, Internet Engineering Task Force, Tech. Rep. 2724, 1999. [Online]. Available: http://datatracker.ietf.org/doc/rfc2724/

[64] B. Claise and B. Trammell, “Specification of the IP Flow Information eXport (IPFIX) Protocol for the Exchange of Flow Information,” RFC 7011, 2013. [Online]. Available: http://datatracker.ietf.org/doc/draft-ietf-ipfix-protocol-rfc5101bis/

[65] S. D’Antonio, T. Zseby, C. Hence, and L. Peluso, “Flow Selection Techniques,” RFC 7014, Internet Engineering Task Force, Tech. Rep. 7014, 2013. [Online]. Available: http://datatracker.ietf.org/doc/rfc7014/

[66] N. Duffield and C. Lund, “Predicting resource usage and estimation accuracy in an IP flow measurement collection infrastructure,” in *Proceedings of the 2003 ACM SIGCOMM conference on Internet measurement - IMC ’03*. New York, New York, USA: ACM Press, Oct. 2003, p. 179. [Online]. Available: http://dl.acm.org/citation.cfm?id=948205.948228

[67] G. Sadasivan, N. Brownlee, B. Claise, and J. Quittek, "Architecture for IP Flow Information Export," RFC 5470, 2009. [Online]. Available: http://datatracker.ietf.org/doc/rfc5470/

[68] A. Kobayashi, B. Claise, G. Muenz, and K. Ishibashi, "IP Flow Information Export (IPFIX) Mediation: Framework," RFC 6183, 2011. [Online]. Available: http://datatracker.ietf.org/doc/rfc6183/

[69] J. Quittek, S. Bryant, B. Claise, P. Aitken, and J. Meyer, "Information Model for IP Flow Information Export," RFC 5102, 2008. [Online]. Available: http://datatracker.ietf.org/doc/rfc5102/

[70] N. Duffield, D. Chiou, B. Claise, A. Greenberg, M. Grossglauser, and J. Rexford, "A Framework for Packet Selection and Reporting," *IETF RFC 5474*, 2009. [Online]. Available: http://datatracker.ietf.org/doc/rfc5474/

[71] B.-Y. Choi and S. Bhattacharyya, "Observations on Cisco sampled NetFlow," *ACM SIGMETRICS Performance Evaluation Review*, vol. 33, no. 3, p. 18, Dec. 2005. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1111572.1111579

[72] B.-Y. Choi, J. Park, and Z.-L. Zhang, "Adaptive random sampling for load change detection," *ACM SIGMETRICS Performance Evaluation Review*, vol. 30, no. 1, p. 272, Jun. 2002. [Online]. Available: http://dl.acm.org/citation.cfm?id=511399.511376

[73] J. Giertl, J. Baca, F. Jakab, and R. Andoga, "Adaptive sampling in measuring traffic parameters in a computer network using a fuzzy regulator and a neural network," *Cybernetics and Systems Analysis*, vol. 44, no. 3, pp. 348–356, 2008. [Online]. Available: http://dx.doi.org/10.1007/s10559-008-9005-0

[74] Q. Xin, L. Hong, and L. Fang, "A Modified FLC Adaptive Sampling Method," in *2009 WRI International Conference on Communications and Mobile Computing*, vol. 2. IEEE, Jan. 2009, pp. 515–520. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4797177

[75] G. Muenz, B. Claise, and P. Aitken, "Configuration Data Model for the IP Flow Information Export (IPFIX) and Packet Sampling (PSAMP) Protocols - RFC6728," IETF RFC 6728, Tech. Rep., 2012. [Online]. Available: http://datatracker.ietf.org/doc/rfc6728/?include_text=1

[76] V. Castro, P. Carvalho, and S. R. Lima, "A cooperative network monitoring overlay," in *Smart Spaces and Next Generation Wired/Wireless Networking*. Springer, 2011, pp. 475–486. [Online]. Available: http://link.springer.com/chapter/10.1007/978-3-642-22875-9_43

[77] J. Schad, J. Dittrich, and J.-A. Quiané-Ruiz, "Runtime Measurements in the Cloud: Observing, Analyzing, and Reducing Variance," *Proc. VLDB Endow.*, vol. 3, no. 1-2, pp. 460–471, Sep. 2010. [Online]. Available: http://dx.doi.org/10.14778/1920841.1920902

[78] A. Pras and J. Schoenwaelder, “On the Difference between Information Models and Data Models - RFC 3444,” IETF, Tech. Rep., 2003. [Online]. Available: https://datatracker.ietf.org/doc/rfc3444/

[79] B. Claise, B. and Trammel, “Information Model for IP Flow Information Export (IPFIX) - RFC 7012,” IETF, Tech. Rep., 2013. [Online]. Available: https://datatracker.ietf.org/doc/rfc7012/

[80] G. Dietz, T and Claise, B and Aitken, P and Dressler, F and Carle, “Information Model for Packet Sampling Exports,” IETF RFC 5477, Tech. Rep., 2009. [Online]. Available: https://datatracker.ietf.org/doc/rfc5477/

[81] “IP Flow Information Export (IPFIX) Entities,” 2015. [Online]. Available: http://www.iana.org/assignments/ipfix/ipfix.xhtml

[82] T. Dietz, B. Claise, and J. Quittek, “Definitions of Managed Objects for Packet Sampling,” RFC 6727, 2012. [Online]. Available: http://datatracker.ietf.org/doc/rfc6727/

[83] B. Case, J and Mundy, R and Partain, D and Stewart, “Introduction and Applicability Statements for Internet-Standard Management Framework - RFC 3410,” IETF, Tech. Rep., 2002. [Online]. Available: https://datatracker.ietf.org/doc/rfc3410/

[84] J. Aitken, P and Claise, B and McDowall, C and Schoenwaelder, “Exporting MIB Variables using the IPFIX Protocol draft-ietf-ipfix-mib-variable-export-09,” IETF, Tech. Rep., 2015. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-ipfix-mib-variable-export/

[85] K. McCloghrie, J. Seligson, F. Reichmeyer, A. Smith, and R. Sahita, “Structure of policy provisioning information (SPPI) - RFC 3159,” IETF, Tech. Rep., 2001. [Online]. Available: https://datatracker.ietf.org/doc/rfc3159/

[86] M. Uslar, M. Specht, S. Rohjans, J. Trefke, and J. M. González, *The Common Information Model CIM: IEC 61968/61970 and 62325-A practical introduction to the CIM.* Springer Science & Business Media, 2012, vol. 66.

[87] S. Sezer, S. Scott-Hayward, P. Chouhan, B. Fraser, D. Lake, J. Finnegan, N. Viljoen, M. Miller, and N. Rao, “Are we ready for SDN? Implementation challenges for software-defined networks,” *IEEE Communications Magazine*, vol. 51, no. 7, pp. 36–43, Jul. 2013. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6553676

[88] M. Jarschel, T. Zinner, T. Hohn, and P. Tran-Gia, “On the accuracy of leveraging SDN for passive network measurements,” in *2013 Australasian Telecommunication Networks and Applications Conference (ATNAC).* IEEE, Nov. 2013, pp. 41–46. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6705354

[89] N. Duffield, “Fair sampling across network flow measurements,” *ACM SIGMETRICS Performance Evaluation Review*, vol. 40, no. 1, p. 367, Jun. 2012. [Online]. Available: http://dl.acm.org/citation.cfm?id=2318857.2254800

[90] R. Hofstede, P. Celeda, B. Trammell, I. Drago, R. Sadre, A. Sperotto, and A. Pras, “Flow Monitoring Explained: From Packet Capture to Data Analysis With NetFlow and IPFIX,” *IEEE Communications Surveys & Tutorials*, vol. 16, no. 4, pp. 2037–2064, Jan. 2014. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6814316

[91] B. Claise, A. Johnson, and J. Quittek, “Packet Sampling (PSAMP) Protocol Specifications,” RFC 5476, 2009. [Online]. Available: http://datatracker.ietf.org/doc/rfc5476/

[92] G. Claise, B and Aitken, P and Johnson, A and Muniz, “IP Flow Information Export (IPFIX) Per Stream Control Transmission Protocol (SCTP) Stream - RFC 6526,” IETF, Tech. Rep., 2013. [Online]. Available: https://datatracker.ietf.org/doc/rfc6526/

[93] A. Orebaugh, G. Ramirez, and J. Beale, *Wireshark & Ethereal network protocol analyzer toolkit.* Syngress, 2006.

[94] V. Jacobson and S. McCanne, “libpcap: Packet capture library,” *Lawrence Berkeley Laboratory, Berkeley, CA*, 2009.

[95] S. Alcock, P. Lorier, and R. Nelson, “Libtrace,” *ACM SIGCOMM Computer Communication Review*, vol. 42, no. 2, p. 42, Mar. 2012. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2185376.2185382

[96] D. Boivie, R. and Feldman, N. and Imai, Y. and Livens, W. and Ooms, “Explicit Multicast (Xcast) Concepts and Options - RFC 5058,” IETF, Tech. Rep., 2013. [Online]. Available: https://datatracker.ietf.org/doc/rfc5058/

[97] V. Jacobsen, C. Leres, and S. McCanne, “Tcpdump/libpcap,” 2005.

[98] D. Song, “Dsniff,” 2000. [Online]. Available: http://www.monkey.org/~dugsong/dsniff/

[99] M. Roesch and Others, “Snort: Lightweight Intrusion Detection for Networks.” in *LISA*, vol. 99, no. 1, 1999, pp. 229–238.

[100] A. Ornaghi and M. Valleri, “Ettercap,” 2005. [Online]. Available: https://ettercap.github.io/ettercap/

[101] C. Inacio and B. Trammell, “YAF: Yet Another Flowmeter.” in *LISA*, 2010.

[102] K. Fujii, “Jpcap–a Java library for capturing and sending network packets,” 2007. [Online]. Available: http://sourceforge.net/projects/jpcap/

[103] R. Gad, M. Kappes, and I. Medina-Bulo, “Improving Network Traffic Acquisition and Processing with the Java Virtual Machine,” in *Computers and Communication (ISCC), 2015 IEEE Symposium on.* Larnaca: IEEE, 2015.

[104] C. Shannon, E. Aben, K. Claffy, D. Andersen, and N. Brownlee, “The CAIDA UCSD Anonymized Internet Traces 2014 - equinix-chicago.dirA.20140619-131100.UTC.anon,” Downloaded from http://www.caida.org/data/passive/passive_2014_dataset.xml, Apr. 2014.

[105] B. K. Tanaka, “Monitoring virtual memory with vmstat,” *Linux Journal*, vol. 2005, no. 140, p. 5, 2005.

[106] R. Krishnan, L. Yong, A. Ghanwani, N. So, and B. Khasnabish, “Mechanisms for Optimizing Link Aggregation Group (LAG) and Equal-Cost Multipath (ECMP) Component Link Utilization in Networks - RFC 7424,” IETF, Tech. Rep., 2015. [Online]. Available: https://datatracker.ietf.org/doc/rfc7424/

[107] B. W. Silverman, *Density estimation for statistics and data analysis*. CRC press, 1986, vol. 26.

[108] ONF, “OpenFlow Switch Specification v1.4.0,” Open Networking Foundation, Tech. Rep., 2013.