UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

NICOLAS SILVEIRA KAGAMI

CAPEST: Network Capacity and Available Bandwidth Estimation in the Data-Plane

Thesis presented in partial fulfillment
of the requirements for the degree of
Master of Computer Science

Advisor: Prof. Dr. Luciano Paschoal Gaspary

Porto Alegre
Setembro 2018

CIP — CATALOGING-IN-PUBLICATION

Kagami, Nicolas Silveira

CAPEST: Network Capacity and Available Bandwidth Estimation in the Data-Plane / Nicolas Silveira Kagami. – Porto Alegre: PPGC da UFRGS, 2018.

64 f.: il.

Thesis (Master) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2018. Advisor: Luciano Paschoal Gaspary.

1. Data plane programmability. 2. Network measurement. 3. Available bandwidth estimation. 4. P4. 5. Packet dispersion. 6. Capacity estimation. I. Gaspary, Luciano Paschoal. II. Título.

“An original idea. That can’t be too hard. The library must be full of them”

— STEPHEN FRY

ACKNOWLEDGEMENT

I would like to thank my family for their love and support throughout my life.

I owe Marina Rey and her family my utmost gratitude, for lovingly welcoming me into their lives.

I thank all my coworkers at Netmetric throughout the years, for treading a long path with me.

I thank my colleague Roberto Costa Filho, for all the help I received throughout my master's degree.

I would like to thank my advisor, Luciano Paschoal Gaspary, for sharing a bit of his wisdom with me.

ABSTRACT

The measurement of available bandwidth and capacity represents an essential requirement for a multitude of network applications spanning from traffic engineering and admission control to network security. Measurement techniques frequently presume to know capacity *a priori*, but this constitutes a weak premise in a number of modern scenarios due to conditions such as abstractions in infrastructure virtualization, dynamic demands in resource sharing and fluctuations in interference, all of which can affect capacity in short time spans. Despite consistent efforts, currently employed techniques struggle to balance accuracy, intrusion and freshness, depending on either substantial intrusion, onerous processing or unfeasible deployment. Recent developments on data plane programmability have breathed new life into this undertaking, allowing observation points to be more efficiently distributed and programmable packet methods to be executed *in-situ*. This dissertation describes CAPEST, a passive capacity and available bandwidth measurement method for the data plane, employing packet dispersion and autocorrelation. The method is evaluated regarding its parametrization sensitivity, its intrusion and freshness in comparison to state-of-the-art techniques and its performance in the real-world application of video routing. CAPEST was found to incur substantially (80%) less intrusion and achieve 10% better accuracy, all the while providing an order of magnitude improvement in freshness.

Keywords: Data plane programmability. network measurement. available bandwidth estimation. P4. packet dispersion. capacity estimation.

Estimando Capacidade e Banda Residual no Plano de Dados

RESUMO

A medição de banda disponível e capacidade representa um requisito essencial para uma multitude de aplicações de rede, abrangendo desde engenharia de tráfego e controle de admissão até segurança de rede. Nesse quesito, técnicas de medição frequentemente presumem conhecer a capacidade *a priori*, mas essa pode ser considerada uma premissa fraca considerando diversos cenários modernos devido a condições como abstrações na virtualização de infraestrutura, demandas dinâmicas no compartilhamento de recursos e flutuações em interferência, todas as quais podem afetar a capacidade em curtos espaços de tempo. Apesar de esforços recorrentes, as técnicas de medição atualmente empregadas ainda encontram dificuldade para balancear acurácia, intrusão e frescor, recaindo em intrusão substancial, processamento oneroso ou implantação inviável. Desenvolvimentos recentes em programabilidade no plano de dados tem dado nova vida a esse esforço, permitindo que pontos de observação sejam mais eficientemente distribuídos e metódos programáveis de pacotes sejam executados *in-situ*. Neste documento apresentamos CAPEST, um método passivo de medição de capacidade e available bandwidth no plano de dados, empregando dispersão de pacotes e autocorrelação. O método é avaliado a respeito da sensibilidade de parametrização, sua intrusão e frescor em comparação a técnicas do estado-da-arte e seu desempenho em uma aplicação realística de roteamento de vídeo. CAPEST permitiu uma redução de 80% em intrusão e um aumento em 10% em acurácia, ao mesmo tempo aprimorando frescor em uma ordem de magnitude.

Palavras-chave: Programabilidade no Plano de Dados, Medição de Rede, Estimativa de Capacidade, Estimativa de Banda Residual.

LIST OF ABBREVIATIONS AND ACRONYMS

ABW	Available Bandwidth
ASIC	Application-Specific Integrated Circuit
BTC	Bulk Transfer Capacity
CM	Capacity Mode
CT	Cross Traffic
DARPA	Defense Advanced Research Projects Agency
DoS	Denial of Service
DSL	Domain Specific Language
FPGA	Field Programmable Gate Array
IDS	Intrusion Detection System
IETF	Internet Engineering Task Force
INT	In-Band Network Telemetry
IP	Internet Protocol
NIC	Network Interface Controller
NSH	Network Service Header
ONF	Open Networking Foundation
OP	Observation Point
PRM	Probe Rate Model
PGM	Probe Gap Model
QoS	Quality-of-Service
QoE	Quality-of-Experience
SDN	Software-Defined Networking
SNMP	Simple Network Management Protocol
SNR	Signal-to-Noise Ratio

TCP	Transmission Control Protocol
TE	Traffic Engineering

LIST OF FIGURES

LIST OF TABLES

CONTENTS

1 INTRODUCTION

Measuring available bandwidth and capacity is vital for several types of network applications. Activities such as traffic engineering, Quality-of-Service (QoS) management (PAUL; TACHIBANA; HASEGAWA, 2016), admission control (CAVUSOGLU; ORAL, 2014), resource provisioning and even network security (GUERRERO; LABRADOR, 2010) depend on the knowledge of either or both available bandwidth and capacity. Link capacity is sometimes considered to be static and known *a priori*, but this premise cannot be guaranteed in an ever increasing number of scenarios. Currently, capacities fluctuate on mobile (MICHELINAKIS et al., 2016) and radio networks as a consequence of variations in range, interference, fading and load (PAKZAD; PORTMANN; HAYWARD, 2015). Also, it has recently been suggested that optical links should adapt their capacities according to Signal-to-Noise-Ratio (SNR) (SINGH et al., 2017). Finally, recent trends of infrastructure virtualization can present dynamic capacities on account of resource sharing and conceal this underlying property due to abstraction. Thus, it is imperative to have a timely and reliable way of gauging capacity-related metrics without exhausting the network resources.

In order to obtain these valuable indicators dynamically, network providers have been relying on *active methods* (PAUL; TACHIBANA; HASEGAWA, 2016; PAKZAD; PORTMANN; HAYWARD, 2015; KIM; LEE, 2014), which consist of disturbing the network in a specific way and observing its behavior (MORTON, 2016). One of the main advantages of this method is the ease of deployment, especially when combined with an end-to-end measurement strategy. In such case, it is possible to cover an entire path, regardless of its size, merely by positioning Observation Points (OPs) at the endpoints (FILHO et al., 2016). However, technologies such as Content Delivery Networks (CDNs), Fog Computing, and 5G device-to-device communications introduce a multitude of possible paths between a given source and destination. The context of ultra-dense networks requires the use of hop-by-hop techniques, which have the ability to infer the performance of a path based on the composition of its links (FILHO et al., 2018). Although this approach solves the problem of scalability, a major problem persists: *active methods* consume the very resources they are trying to measure.

In order to solve the overhead problem, network operators can rely on *passive methods*, which derive the same metrics from preexisting packets (JULURI; TAMARAPALLI; MEDHI, 2016; MEGYESI et al., 2017; MICHELINAKIS et al., 2016; KATTI et al., 2004). This approach mitigates the problem of overhead as they measure live traffic. However, they demand a more significant effort in filtering and analysis to be fresh and accurate. Several *passive methods* require the gathering of traces from the network and relegate this burden to the control plane. In addition, current implementations of such approaches have a complex deployment since they either require the installation of dedicated equipment at each OP or risk degrading the performance of the live traffic by burdening the network devices with CPU intensive tasks.

Despite current efforts, network measurement applications still struggle to balance accuracy, intrusiveness and freshness for optimal management. This is the very problem we tackle in this dissertation. Current strategies either introduce a substantial amount of intrusiveness or lack freshness or require cumbersome deployment to work well in large-scale networks. Fortunately, recent developments on data plane programmability have brought about new possibilities of navigating the solution space of network measurement. Generic switch programmability allows packet methods to be executed in-situ, which has the potential to alleviate intrusiveness and provide fresh information. Furthermore, the ability to strategically distribute measurement vantage points can also contribute to attaining a complete picture of the network without resorting to detereorating accuracy, freshness or overhead.

In this dissertation, we propose a passive method for estimating link capacity and available bandwidth from the data plane, henceforth refered to as CAPEST (Capacity and Available Bandwidth Estimation method). The measurement method is based on packet dispersion, making use of a bin histogram to quantify dispersion occurrence, filtering outliers by a reverse estimate autocorrelation. CAPEST collects information about packet length and timing from live packets employing *in-situ* packet methods developed in the data plane via P4. The analysis is triggered by a special probe packet, which also carries the result at line rate via In-Band Network Telemetry, improving freshness and reducing intrusiveness. CAPEST's per-hop measurement granularity allows efficient estimation of path properties through link composition. Finally, this dissertation's contributions can be summarized as follows.

- **Cost-effective data plane measurement method.** We propose a capacity and available bandwidth measurement method capable of balancing the essential design requirements of accuracy, intrusiveness, freshness and deployability.

- **Comprehensive evaluation.** We evaluate how CAPEST's parameters influence its performance, compare its accuracy, freshness and overhead to state-of-the-art solutions and assess its applicability to video routing.
- **Data plane programmability insights.** We explore a novel technology within a restrictive framework and present key insights into the challenges and opportunities faced when applying it to a complex mechanism.

The remainder of this document is organized as follows. In Chapter 2, we introduce background concepts of network programmability, data plane programmability and the state-of-the-art regarding the network measurement techniques of available bandwidth and capacity. Then, in Chapter 3, we outline the conceptual building blocks upon which CAPEST is built, presenting metrics, design objectives, architecture and measurement method. In Chapter 4, we detail how the proposed method is implemented in the framework of data plane programmability in regards to algorithm parametrization, data structures and packet triggers. In Chapter 5, we describe how CAPEST was evaluated, encompassing sensitivity analysis, cost comparison and a real-world application performance. In Chapter 6, we describe and discuss lessons learned about developing complex applications for P4. Finally, in Chapter 7, we summarize the dissertation's contribution, along with a perspective for future work.

2 BACKGROUND AND STATE OF THE ART

In this chapter we describe the most important concepts and approaches that constitute the state of the art regarding network programmability, data plane programmability and the techniques for measuring available bandwidth and capacity.

2.1 Network Programmability

Historically, one of the most challenging activities within network management has been the deployment and configuration of network policies. In order to implement these policies, operators are often required to configure devices individually, frequently involving low-level and vendor-specific commands. Effectively, this rigidity hinders the network's adaptability to events such as faults and load fluctuations, considering automatic reconfiguration mechanisms are very difficult to be deployed under these circumstances (KREUTZ et al., 2015). This issue has been more generally dubbed "ossification" by the research community and has spurred a series of efforts from both the academy and the industry ever since.

2.1.1 Active Networks

In the late 90s, the emergence of Active Networks represented one of the earliest notable initiatives toward network programmability, primarily backed by the Defense Advanced Research Projects Agency (DARPA). This endeavor focused on bringing programmability to network nodes, embracing a couple of approaches, according to the programming model:

- **Programmable Capsule model:** instructions and information carried in-band via data or dedicated packets, *e.g.*, ANTS capsule-based system (WETHERALL; GUTTAG; TENNENHOUSE, 1998);
- **Programmable Switch model:** switch configurations (for packet methods) installed via out-of-band management, *e.g.*, SwitchWare (SMITH et al., 1996).

Active Networks sought to replace the traditional concept of simple forwarding nodes with abstractions to expose the node's resources and functions, envisioning applications for a variety of up-and-coming network applications, such as congestion control, reliable multicasting and active caching (TENNENHOUSE et al., 1997). Ultimately, Active Networks failed to gain widespread adoption mainly due to the lack of a clear path of deployment, but would serve to inspire a substantial amount of following efforts (FEAMSTER; REXFORD; ZEGURA, 2014).

2.1.2 Control Plane Separation

In the early 2000s, network operators struggled to deal with the increase in traffic bandwidth and volume, pushing for new approaches to traffic engineering (TE) that would be able to manage the network with flexibility, consistency and performance. In response to this issue, and along with improvements to servers' memory capacity and processing power, researchers focused on developing deployable and practical solutions, culminating in two major trends:

- an **open and standardized interface to the data plane** served to abstract and decouple design constraints, leading to the development of innovative ideas on both the control and data plane. In this aspect, the Internet Engineering Task Force (IETF) spearheaded the Forwarding and Control Element Separation (ForCES) (YANG et al., 2004) and the Netlink interface to Linux Kernel (SALIM et al., 2003);
- a **logically centralized control** allowed network applications to have a network-wide view, helping to coordinate efforts more effectively, most notably exemplified by the Routing Control Platform (RCP) (CAESAR et al., 2005). This trend brought advantages such as more adaptability, better load balancing and more thorough security measures.

Despite a more pragmatic approach than its Active Networks predecessors, the adoption of APIs such as ForCES remained feeble, mainly on account of the lack of backward compatibility, discouraging vendors from conceding their market share (FEAMSTER; REXFORD; ZEGURA, 2014). On the other hand, RCP reused the Border Gateway Protocol (BGP) (REKHTER; LI; HARES, 2005) to install forwarding tables, but its scope left little room for innovation outside of routing management. Eventually, these difficulties would serve as learning experience and influence the specification of OpenFlow (MCKEOWN et al., 2008).

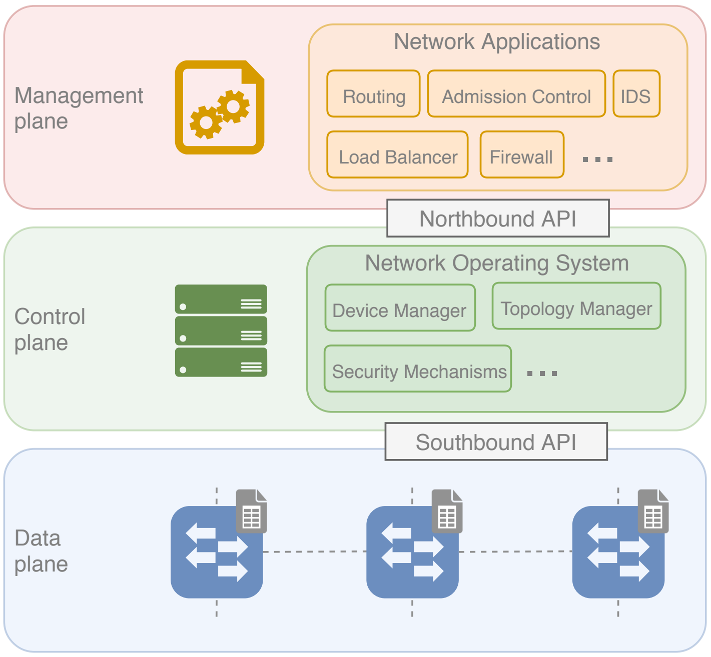
2.1.3 Software-Defined Networking

Another group of approaches expanded on the concept of control plane separation, establishing entirely new architectures for network control and management. The 4D approach suggested four different planes, dividing into *decision* (to translate network objectives into packet-handling states), *dissemination* (to install the forwarding rules), *discovery* (to collect topology and measurements) and *data* (to enforce the forwarding rules). Building upon this "clean slate" paradigm, the Secure Architecture for Networked Enterprise (SANE) (CASADO et al., 2006), and its successor Ethane (CASADO et al., 2007), developed a centralized control enterprise solution, where switches were reduced to flow tables populated according to high-level policies.

In the wake of these contributions, by 2008, Mckeown *et al.* cemented the Software-Defined Networking paradigm with the release of OpenFlow (MCKEOWN et al., 2008). OpenFlow elaborated on the simple switch design proposed by Ethane, defining the switches (or forwarding elements) as a pipeline of flow tables with a (1) matching rule for the packets, (2) actions to be taken on match and (3) corresponding counters to keep statistics on the matching rule (KREUTZ et al., 2015). Its ability to coexist with traditional devices allowed it to be progressively adopted by switch vendors, being comprehensive in scope and compatible in deployment. Crucially for its success, OpenFlow trod the line between conceptual and practical, eventually becoming (along with SDN) the first widely adopted approach in network programmability. This success led to (1) the foundation of the Open Networking Foundation (ONF), helping to promote open technologies in the marketplace and establishing open networking standards; and (2) the advancement of the SDN architecture.

Currently, the SDN architecture can be described in three different planes, as can be seen in Figure 2.1. From a bottom-up perspective, the data plane is composed of simple forwarding devices, executing match-action rules according to a forwarding table. The data plane communicates with the control plane via the southbound API. This layer is critical to the concept of separation of the control plane and is where OpenFlow most established its contribution.

Figure 2.1: The software-defined networking architecture defines three planes with different attributions.



Source: Adapted from Software-Defined Networking: A comprehensive survey *et al.* (KREUTZ et al., 2015).

The southbound API is comprised of communication standards such as the event-based messages to inform topology changes, the collection of statistics and the *packet-in* message, where the controller helps the forwarding device set up a new flow. The OpenFlow protocol establishes the most acclaimed example of a southbound API.

At the heart of the control plane lies the Network Operating System (NOS), also known as the controller, which serves to bridge the gap between the low-level devices and the high-level policies. The NOS is responsible for an assortment of tasks aimed at facilitating the network management, from providing abstractions and managing resource utilization to device discovery and security mechanisms (KREUTZ et al., 2015).

The northbound API was conceived as a way to improve the deployment of network applications by offering a controller-independent interface. Unlike the southbound API, the northbound API has not yet established a *de facto* standard such as OpenFlow, but a number of approaches have been proposed, such as RESTful APIs, multilevel programming interfaces and *ad hoc* APIs. Finally, the management plane houses the applications that implement network control and functionalities via the northbound API. These applications include services like routing, load balancing and firewalls, determining the policies operating on the network.

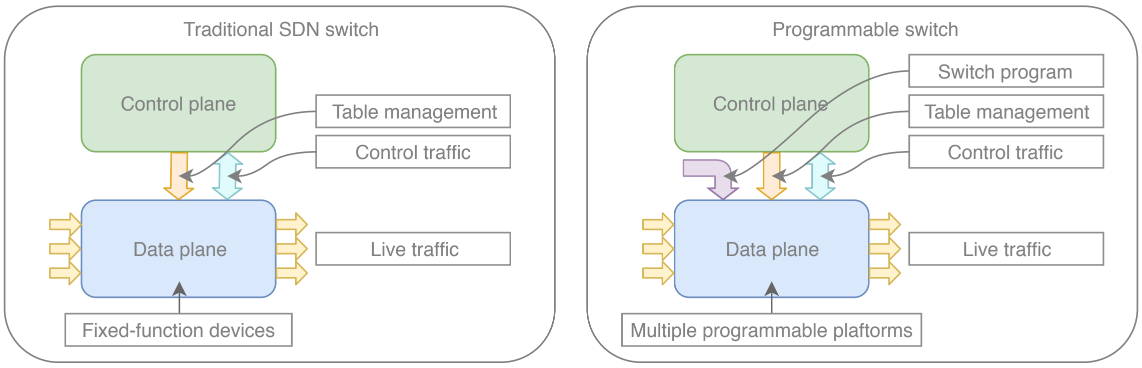
2.2 Data Plane Programmability

With the advent of SDN, attention was turned toward the advantages of dynamically programming the forwarding rules of the data plane. In this context, an issue became more pronounced: switch functionality was still "ossified", being mostly defined by inbuilt design. More specifically, the set of functions and features offered by the data plane remained hostage to the long product cycles and the static nature of fixed-functions ASICs. Operators found themselves incapable of programming custom packet methods and headers, hampering the development of novel network protocols in times of change. Starting in the late 2000s, a few researchers addressed this problem by attempting to redesign the data plane to more adequately match the flexibility of the control plane (ANWER; FEAMSTER, 2009) (BOSSHART et al., 2013). By the early 2010s, these efforts converged on the idea of data plane programmability (SONG, 2013) (BOSSHART et al., 2014).

Following on the footsteps of general purpose computing, a new layer of abstraction was introduced, making use of the trends of network programmability and overall growth in computing power to considerably improve flexibility and hardware support. With data plane programmability, a programming language can be used to generically describe forwarding functions. Switch programs can be compiled to run on multiple platforms, allowing the network operator to dynamically reconfigure its behavior according to need. Considering the market, competition is encouraged since multiple new technologies are available to perform the functions of a switch. Additionally, the manufacturers can abstract low-level architectural details from their high-level packet-processing description, allowing the internal switch design to remain undisclosed.

As illustrated in Figure 2.2, data plane programmability represents a small but consequential modification to the traditional SDN architecture. Backward compatibility is maintained since the communication between the data plane and the control plane is not required to change. In fact, protocols like OpenFlow are still necessary to populate forwarding tables and can now benefit from faster device adaptability and dissemination of new features.

Figure 2.2: A comparison between the traditional SDN switch and the programmable switch.



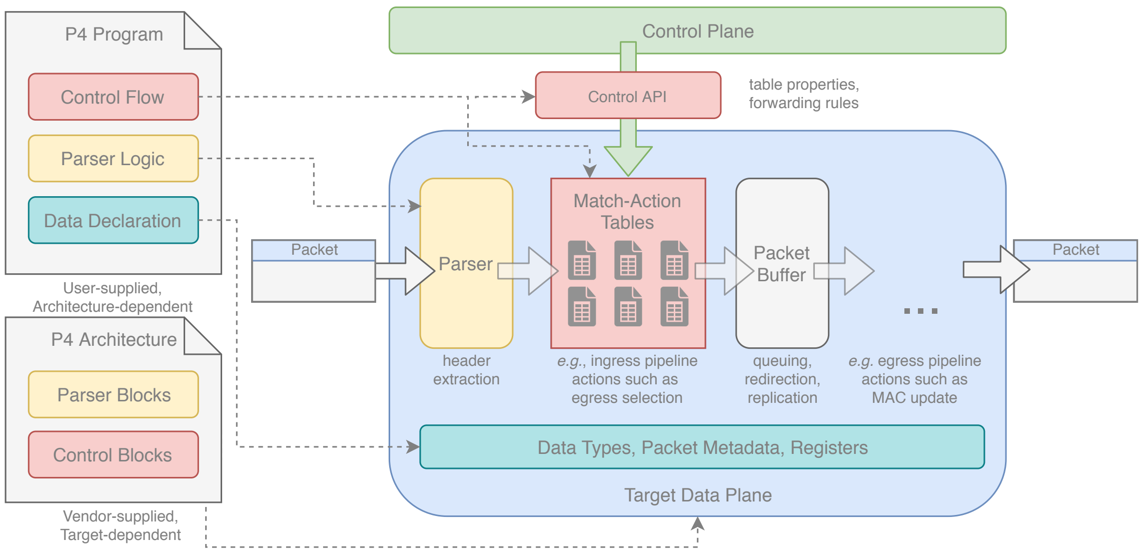
Source: Adapted from the P4 specification (The P4 Language Consortium, 2018).

2.2.1 P4

One of the most fruitful programming languages within this new substrate has been P4, which stands for "Programming Protocol-independent Packet Processor" (BOSSHART et al., 2014). The P4 language provides a high-level description of data plane behavior, designed to be compatible with several different physical targets such as software switches, Network Interface Cards (NICs), Field Programmable Gate Arrays (FPGAs) and Application-Specific Integrated Circuits (ASICs). In order to achieve this level of support, each target requires a (1) dedicated architecture definition, (2) software implementation framework and (3) P4 compiler. After a manufacturer has provided the aforementioned items, a P4 program can be written for that target, establishing the data plane forwarding logic and the set of tables and objects available for management from the control plane. The P4 language can be subdivided into four distinct components: (1) architecture definition, (2) data declaration, (3) parsing logic and (4) control flow (The P4 Language Consortium, 2018).

The **target architecture** is a separate component from the P4 program which defines the programmable blocks and internal structure of the packet processor. This component is how the manufacturer exposes their switches' features to its users. The final target can be composed of two types of programmable blocks: *parser blocks* and *control blocks*, which are connected within *packages*. The architecture may also define helpful data types, constants, errors and other implementations (The P4 Language Consortium, 2018). The content of parser and control blocks is determined by the other components in the P4 program, as illustrated by Figure 2.3.

Figure 2.3: The P4 program and architecture and their mapping to the P4 abstract forwarding model.


P4 Program
Control Flow
Parser Logic
Data Declaration
User-supplied, Architecture-dependent
P4 Architecture
Parser Blocks
Control Blocks
Vendor-supplied, Target-dependent
Control Plane
Control API
table properties, forwarding rules
Packet
Parser
Match-Action Tables
Packet Buffer
...
Packet
header extraction
e.g., ingress pipeline actions such as egress selection
queuing, redirection, replication
e.g. egress pipeline actions such as MAC update
Data Types, Packet Metadata, Registers
Target Data Plane

Source: Adapted from multiple sources (BOSSHART et al., 2014) (CORDEIRO; MARQUES; GASPARY, 2017) (The P4 Language Consortium, 2018).

Within the P4 program, the **data declaration** segment can determine and instantiate header formats, data types, errors and variables, which are made available throughout the control blocks. Header formats, in particular, are defined as a set of fields and their sizes, defining the basic building blocks for the parsing procedure.

The **parsing logic** defines a state machine responsible for classifying each incoming packet into one of two states: acceptance or rejection. Each parser state can be composed of statements and references to other states. In this stage, packets are only accepted if their structure (*e.g.*, headers) conforms to a recognized pattern (*e.g.*, Ethernet, IP, TCP). Ultimately, the packet fields that conform to the parsing states are extracted and made available to subsequent control blocks as metadata.

Control flows express the computations carried out using match-action tables, attributed to the control blocks. As the name implies, this component also defines when and how table actions are to be invoked. While the tables are populated by the control plane, the control flows configure how the tables are exposed to the control plane (via the control API) and how the actions affect the packet metadata and headers (*e.g.*, determining egress port and updating MAC addresses).

From its inception, P4 has attracted attention from both the industry and the academia alike, gaining support from the ONF and the Linux Foundation. P4's ability to express a wide variety of packet manipulations has helped bring new network functionality, previously performed by middleboxes, to the data plane. The community has developed projects from firewalls and encapsulation to load balancing (KATTA et al., 2016), heavy-hitter detection (SIVARAMAN et al., 2017) and in-band network telemetry (KIM et al., 2015).

2.2.2 In-Band Network Telemetry

The trend of bringing functionality to the data plane has allowed many facets of network management to be revisited, particularly in telemetry. Traditional network management methods often rely on the client-server model (or "pull model"), in which a main device periodically requests and collects information from the other entities. This centralization entails substantial limitations on scalability, especially when combined with the vigorous increase in the number of elements in the network. Additionally, traditional telemetry solutions do not allow a precise view of how a specific packet experiences the network.

In order to tackle these limitations, In-band Network Telemetry (INT) was suggested by the P4 community as a means of extracting and exporting information directly from the data plane (Changhoon Kim et al., 2016). This approach is not without precedent. Harking back to the efforts of programmable capsules in Active Networks, Smart Packets had expressed similar intentions in 1999 (SCHWARTZ et al., 1999), suggesting the distribution of management decision points and the collection of specific information locally, instead of via polling like with the Simple Network Management Protocol (SNMP). Another contribution in the history of INT was Tiny-Packet-Programs (TPP), which proposed simple programs carried by packets to be executed on switches, envisioning applications such as congestion control, measurement and verification (JEYAKUMAR; ALIZADEH; GENG, 2014).

Essentially, INT proposes a framework for embedding telemetry instructions and data into packet header fields. There is not a unique encapsulation standard, instead, the INT specification suggests a few compatible protocols (*e.g.*, NSH, VXLAN, Geneve), which can present different advantages. In this framework there are three roles a network device can perform:

- the **source** creates and inserts INT headers into its packets, determining which instructions are to be executed in the following hops;
- the **transit hop** interprets the INT instructions and adds the respective metadata to the packet; and
- the **sink** extracts and consumes the INT instructions and metadata, removing the INT header to make INT transparent to upper layers.

These INT-enabled network elements effectively code the state of the network into transiting packets, allowing for high-level applications such as congestion control, routing and verification (Changhoon Kim et al., 2016).

There are many advantages to INT, such as the placement of vantage points at switches to empower network measurements (MOSHREF et al., 2016). In this picture, a single packet can collect data from all compatible nodes in its path in one trip, amortizing the required intrusiveness and disseminating the information at the same time. It can also enable switch-local decisions (*e.g.*, decisions made entirely between neighboring switches), without having to resort to control plane intervention. Most importantly, the capacity to locally produce diagnostic information can help network management applications to be implemented more efficiently than a centralized model.

2.3 Network Measurement Methods

Considering the potential of new programmable paradigms in computer networks, it is interesting to revisit more traditional problems such as network measurement. In recent years, several investigations have proposed different techniques for estimating network capacity and available bandwidth (ABW). Despite significant advances, a truly efficient technique for jointly measuring these two metrics remains an open question. This section classifies these investigations into three different axes, presented as follows. The first axis examines how existing techniques differ according to the method used for network measurement (passive or active). The second axis evaluates the related work by analyzing the scope of measurement (end-to-end or hop-by-hop). Finally, the third axis encompasses recent work that takes advantage of Software-Defined networks to introduce new measurement approaches.

Table 2.1: State-of-the-art approaches for available bandwidth and capacity estimation.

Method Name	Method Type	Scope	SDN	DPP	Base Technique	ABW	Capacity
CAPEST	Passive	Hop	Yes	Yes	Packet Dispersion	Measures	Measures
Megyesi	Passive	Hop	Yes	No	Utilization Monitoring	Measures	Requires
(MEGYESI et al., 2017)							
Next-V2	Active	Path	No	No	Probe Rate Model	Measures	N/A
(PAUL; TACHIBANA; HASEGAWA, 2016)							
Michelinakis	Passive	Hop	No	No	Packet Dispersion	N/A	Measures
(MICHELINAKIS et al., 2016)							
Pakzad	Active	Hop	Yes	No	Packet Dispersion	N/A	Measures
(PAKZAD; PORTMANN; HAYWARD, 2015)							
Zhang	Active	Path	No	No	Packet Dispersion	N/A	Measures
(ZHANG; XU, 2015)							
SigMon	Active	Path	No	No	Probe Gap Model	Measures	Measures
(KIM; LEE, 2014)							
Ohkawa	Passive	Path	No	No	Packet Dispersion	N/A	Measures
(OHKAWA; NOMURA, 2014)							
EKF-UI	Passive	Path	No	No	Adaptive Kalman Filter	Measures	Requires
(CAVUSOGLU; ORAL, 2014)							
MultiQ	Passive	Hop	No	No	Equally-Spaced Mode Gaps	N/A	Measures
(KATTI et al., 2004)							
CapProbe	Passive	Path	No	No	Packet Dispersion	N/A	Measures
(KAPOOR et al., 2004)							
PathChirp	Active	Path	No	No	Probe Rate Model	Measures	N/A
(RIBEIRO et al., 2003)							
Spruce	Active	Path	No	No	Probe Gap Model	Measures	Requires
(STRAUSS; KATABI; KAASHOEK, 2003)							
PathLoad	Active	Path	No	No	Probe Rate Model	Measures	N/A
(JAIN; DOVROLIS, 2002)							
PathRate	Active	Path	No	No	Packet Dispersion	N/A	Measures
(DOVROLIS; RAMANATHAN; MOORE, 2001)							
TOPP	Active	Path	No	No	Probe Rate Model	Measures	N/A
(MELANDER; BJORKMAN; GUNNINGBERG, 2000)							
PathChar	Active	Hop	No	No	Variable Packet Size	N/A	Measures
(DOWNEY, 1999)							

Source: The Author.

2.3.1 Active vs Passive Approaches

As shown in Table 2.1, measurement techniques can be categorized into active and passive methods, which present a complementary nature in aspects such as accuracy and resource utilization. Active methods make use of dedicated packets streams by injecting an artificial load into the network and accurately measuring its properties (MORTON, 2016). One common active approach is known as the Probe Rate Model (PRM), employed by techniques such as Next-V2 (PAUL; TACHIBANA; HASEGAWA, 2016), Assolo (GOLDONI; ROSSI; TORELLI, 2009), PathChirp (RIBEIRO et al., 2003), TOPP (MELANDER; BJORKMAN; GUNNINGBERG, 2000) and PathLoad (JAIN; DOVROLIS, 2002). In this approach, packet probes are sent at ever-increasing rates, while measuring the arriving rate at the end-point. ABW is detected when congestion is induced, *i.e.*, the arriving rate becomes lower than the sending rate. Since PRM relies on congestion to measure ABW, it inevitably disturbs the network with overhead.

A less intrusive active approach is the Probe Gap Model (PGM), which is the basis for methods such as Sigmon (KIM; LEE, 2014) and Spruce (STRAUSS; KATABI; KAASHOEK, 2003). This approach estimates ABW indirectly from the cross traffic observed in the temporal dispersion between consecutive probes and requires previous knowledge of capacity (MICHAUT; LEPAGE, 2005). Despite their relatively lower intrusiveness, their overhead remains orders of magnitude higher than that of passive methods. Additionally, low intrusiveness active methods present a critical problem: they are prone to measure an artificially reduced capacity. It turns out that most wireless networks, such as 4G and upcoming 5G networks, rely on dynamic resource allocation algorithms (MARTÍN-SACRISTÁN et al., 2009). In such a shared and dynamic environment, the amount of resources allocated for a given user is directly proportional to the network load generated by this user. Therefore, a low intrusiveness method will probably measure an artificially reduced capacity, which would be significantly higher if the method required more resources.

On the other side of the spectrum, passive methods may be better suited to observe short-term and dynamic behaviors(CAVUSOGLU; ORAL, 2014). For example, Ohkawa (OHKAWA; NOMURA, 2014), EKF-UI (CAVUSOGLU; ORAL, 2014), CapProbe (KAPOOR et al., 2004) rely on statistical methods, such as Packet Dispersion and Adaptive Kalman Filters, to analyze live packets and derive the desired metrics. Such non-intrusive methods preserve and measure actual traffic, but require a greater effort in filtering and analysis to achieve acceptable accuracy. Crucially, passive methods solve one of the main issues associated with active techniques, namely overhead, since they do not consume the very resources they are trying to measure. However, a crucial problem remains: most of the evaluated methods do not address high-scale deployability, *i.e.*, how to effectively distribute the OPs (OHKAWA; NOMURA, 2014; CAVUSOGLU; ORAL, 2014; KATTI et al., 2004; KAPOOR et al., 2004). Additionally, none of the passive methods in Table 2.1 estimates ABW without knowing capacity beforehand, usually justified by outdated assumptions about capacity. This is a major limitation because capacity information is highly dynamic in shared access networks such as 4G/LTE, 5G and Wi-Fi.

2.3.2 End-to-End vs Hop-by-Hop Approaches

Capacity and ABW measurement techniques can be further divided according to measurement scope. End-to-end proposals evaluate the properties of a path between two end-points, recent examples of which include NEXT-V2 (PAUL; TACHIBANA; HASEGAWA, 2016), Pakzad (PAKZAD; PORTMANN; HAYWARD, 2015), Zhang (ZHANG; XU, 2015) and SigMon (KIM; LEE, 2014). These methods frequently obtain information limited to the worst link in the path (*i.e.*, narrow or bottleneck link). Therefore, acquiring a detailed depiction of the network employing exclusively end-to-end measurements can become very burdensome depending on the topology and the distribution of OPs. To that effect, Ohkawa *et al.* (OHKAWA; NOMURA, 2014) argue that their path method can be used to monitor all paths, but do not specify how this deployment can be achieved. There are a few techniques, most notably pathChar (DOWNEY, 1999) and multiQ (KATTI et al., 2004), that attempt to extract multiple per-hop measurements from an end-to-end perspective. These techniques pursue an ambitious concept but do not manage to achieve the realiability required by modern network management, as they require particular conditions to work well (such as not encountering queuing delay on a packet probe) (DOVROLIS; RAMANATHAN; MOORE, 2001) and are not guaranteed to acquire information about all hops.

Considering that the per-hop scope of information can be composed into path estimates (FILHO et al., 2018), a more reliable approach to acquiring network information is to deploy key OPs at the hops. Michelinakis *et al.* (MICHELINAKIS et al., 2016) deploy OPs at the last hop and acquire information concerning the edge of a mobile network. However, measuring the internal nodes is essential for a complete picture of the network.

A comprehensive deployment can be facilitated by SDN (as employed by the works of Megyesi (MEGYESI et al., 2017) and Pakzad (PAKZAD; PORTMANN; HAYWARD, 2015)), which is further explained in the following subsection.

2.3.3 Software-Defined Networking Approaches

Software-Defined Networking (SDN) has introduced a more flexible and dynamic way of programming forwarding paths in complex network environments. The availability of a global view of the network topology through a centralized control plane allows open problems and challenges, such as capacity and ABW estimations, to be revisited. Megyesi *et al.* (MEGYESI et al., 2017) proposed a passive technique that is implemented as an extension of the Netflow protocol. The technique consists of monitoring bandwidth utilization along the network to calculate ABW, relying on prior knowledge of the capacity of each link. Considering that prior knowledge of the link capacities is not guaranteed in today's networks, Pakzad *et al.* (PAKZAD; PORTMANN; HAYWARD, 2015) have suggested a technique for measuring capacity in SDN networks, which makes use of the well-known packet pair technique. However, their method does not encompass ABW estimation. Considering that SDN represents one of the latest frontiers for its advantages to deployment and scalability, we expect it to become more commonplace within measurement techniques in the future. The application of SDN has been successful in the monitoring of other network metrics such as round-trip time (Atary; Bremler-Barr, 2016). In this sense, current developments in data plane programmability also offer a yet unexplored opportunity to implement the network measurement of ABW and capacity.

Given the above, CAPEST contributes a step forward in the state-of-the-art since, unlike the related work, it introduces a novel passive technique with the ability to simultaneously estimate both capacity and ABW. The proposed technique leverages both programmable data plane and in-band network telemetry to provide accurate per-link measurements while incurring minimal overhead. Finally, we advocate this approach is flexible, since the estimated metrics can either be transported via INT and gathered at a centralized decision point, but are also available locally at each hop, enabling local decisions based on the accurate and fresh information.

3 CONCEPTUAL FRAMEWORK

In this chapter, we outline the framework upon which CAPEST is built. We start by formally defining the measurement metrics of interest to CAPEST. We then enumerate and discuss the design requirements of the proposed solution. Finally, we introduce and describe the CAPEST architecture and the associated measurement method.

3.1 Target Metrics

CAPEST's process culminates in the estimation of two metrics: Capacity and Available Bandwidth. In this section, we define these metrics and their relevant characteristics.

3.1.1 Capacity

In concordance with Prosad *et al.* (PROSAD et al., 2003), capacity is defined as the maximum possible transference rate achievable at the network layer. This rate is mostly determined by technical factors of the underlying layers, such as link layer protocol, the bandwidth limitations inherent to the propagation medium and SNR. Since these properties vary from hop to hop, it is convenient to define capacity at a per-hop level of detail. The observed capacity of a path between end-points a and b can be estimated by a composition of its constituent link capacities, limited by the lowest singular value, given by Equation 3.1:

$$C_{a,b} = min(C_a, ..., C_b) \tag{3.1}$$

3.1.2 Available Bandwidth

Available bandwidth is defined as the unused capacity of a link or a path within a time interval (PROSAD et al., 2003). It can also be described as the the amount of traffic that can be introduced to a path without interfering with the traversing flows. Formally, the available bandwidth A_i of a link i can be inferred from its capacity C_i and its average utilization u_i as seen in Equation 3.2. Similarly to capacity, the available bandwidth of a path can be composed as the minimal value of its component links, as given by Equation 3.3.

$$A_i = C_i \cdot (1 - u_i) \tag{3.2}$$

$$A_{a,b} = min(A_a, ..., A_b) \tag{3.3}$$

An accurate assessment of both capacity and available bandwidth is highly descriptive of the network and is fundamental for a myriad of applications. For example, these metrics have been instrumental in rate-distortion algorithms (CAVUSOGLU; ORAL, 2014), admission control (BRESLAU et al., 2000) and QoE-aware route selection (FILHO et al., 2018). Expanding on the latter example, Costa *et al.* employ periodic ABW measurements to estimate path QoE for video streaming applications (since ABW is the most significant QoS indicator of QoE (CASAS et al., 2016; RAMAKRISHNAN et al., 2015)), while implicitly using capacity to model congestion dynamics in route selection (FILHO et al., 2018).

3.2 Design Objectives

In this section, we define the most important design objectives of a network measurement technique, depicting their trade-offs and how different classes of methods excel at these aspects.

3.2.1 Accuracy

One of the most important factors that define a measurement tool is accuracy. A method's accuracy is defined by the proximity of its results to the reference ground truth. Henceforth in this paper, the accuracy of a measurement is determined by the application of Equation 3.4, given a positive measurement value V_M and a positive reference value V_R.

$$Accuracy(V_R, V_M) = 1 - \frac{|V_R - V_M|}{V_R} \tag{3.4}$$

As a design objective for a our measurement method, a percentage error rate of up to one-digit is desired, as is typically reported by comparable state-of-the-art solutions. However, most techniques achieve this level of accuracy at the cost of high resource consumption, either by disturbing the network or laboriously analyzing packet statistics.

3.2.2 Freshness

Freshness is defined as the quality of having recently monitored data available as soon as possible. In this context, two main contributing factors can be identified as the availability of recent data and the timeliness of the measurement process (*i.e.*, the time it takes for the process to produce a measurement). This ability is particularly relevant for network management whenever an event requires a swift response, such as traffic engineering or denial of service (DoS) mitigation. We expect the freshness of our solution to at least rival that of *active methods*, pursuing an optimal timeliness of an RTT. Yet, it should be noted that *active methods* provide freshness by injecting their own statistical fodder into the network, incurring in higher intrusiveness.

3.2.3 intrusiveness

intrusiveness can be described as the degradation of service experienced by production packets. One of the most prevalent indicators of intrusiveness is the amount of data that needs to be introduced by monitoring mechanisms to the network in order to perform measurements. At its mildest, this aspect can increase latency on account of larger queues and, at its worst, it can induce packet loss and reduction of throughput. In the context of our solution, an acceptable amount of intrusiveness is displayed by *passive methods* since they infer metrics by observing the underlying traffic instead of injecting dedicated data streams.

3.2.4 Scope

Measurement scope refers to the spatial granularity of acquired information, *e.g.*, whether a measurement pertains to a complete path or specific link. The direct measurement of a path's properties has the ability to simultaneously consider all of the contributing factors (*e.g.* medium bandwidth, resource sharing, noise) and their interactions along the way. However, in the context of a large-scale network, in which there is a combinatorial amount of possible paths, it quickly becomes an exceedingly impractical endeavour to measure each path individually. Considering that it is much more feasible to employ per-hop measurements from which path estimations can be composed (FILHO et al., 2018), our solution's ideal measurement scope is hop-by-hop.

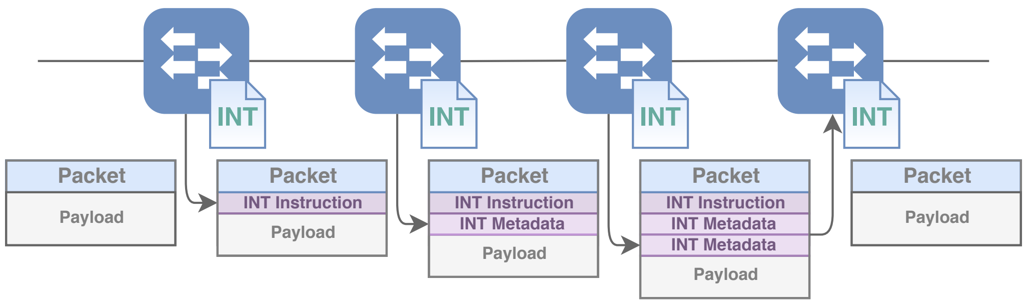
In conclusion, the ideal measurement method would approximate 90% accuracy, the freshness of an *active method* and the resource consumption of a *passive method*, all the while providing a per-hop scope of estimation. This is especially challenging since most of these aspects counteract each other. However, the emergence of data plane programming has the potential to reestablish this paradigm.

3.3 Overall Architecture

CAPEST relies on the existence of programmable switches within the network. Even though this concept of network programmability and intelligent forwarding devices is not a recent aspiration, current developments of generic switch programmability (*e.g.* through P4 (BOSSHART et al., 2014)) have laid fertile ground for the feasible implementation of data plane methods. This approach allows procedures to be generically programmed and dynamically instantiated in network elements, allowing complex and adaptable behaviour to operate at line rate. In this context, methods can be triggered by packets that respect match-action rules and can consume (stateful) information maintained in the switch.

Considering the advantages provided by packet level telemetry (JEYAKUMAR; ALIZADEH; GENG, 2014; ZHU et al., 2015), CAPEST makes use of Inband Network Telemetry (Changhoon Kim et al., 2016), as a means for extracting and exporting information directly from the data plane. Additionally, INT is employed implementing the NSH encapsulation standard in order to export telemetry data from the programmable switches to wherever they are needed. INT-enabled network elements selectively introduce information into transiting packets, effectively coding the state of the network for other elements to consume (*e.g.*, the switches or an external controller), as illustrated by Figure 3.1.

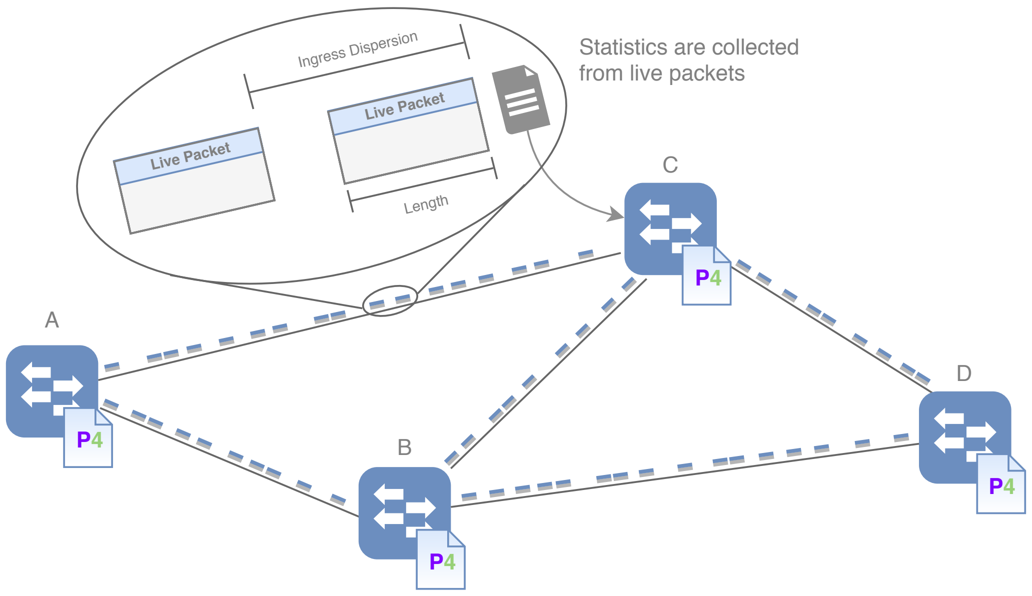
Figure 3.1: INT introduces metadata to packets through header manipulation.



Source: The Author.

Essentially, CAPEST is a loadable module to the data plane of a generically programmable switch. CAPEST makes use of trigger methods in order to collect packet information, process it, and relay to the rest of the network its estimates of capacity and available bandwidth. CAPEST's mechanism differentiates between two kinds of packets, live packets and probe packets.

Figure 3.2: CAPEST collects statistical information from live packets.

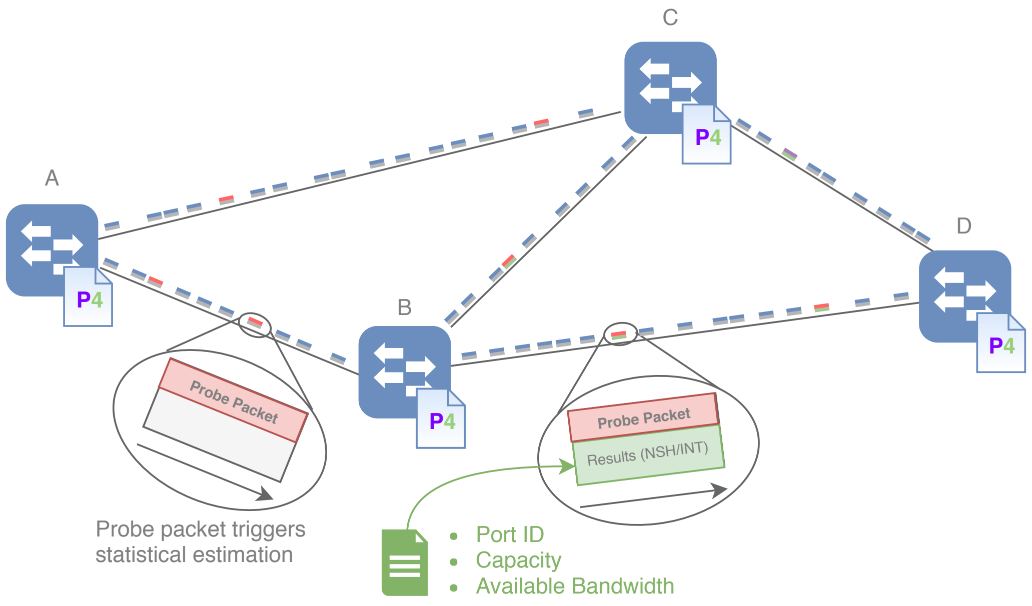


Source: The Author.

Live packets represent the user traffic coursing through the network and are the object of measurement. These packets have external sources and targets, connecting users to utilities and establishing services. In this context, it is imperative that they experience a reliable and efficient traversal. With this in mind, CAPEST's procedures for live packets are mostly limited to collecting information regarding length and packet dispersion, as illustrated by the transiting packets arriving at the "C" switch in Figure 3.2. This procedure can be deployed on a per-packet basis or be coupled with a filter.

Probe packets are artificially inserted into the network whenever estimations are necessary. Unlike live packets, probe packets have a Network Service Header implementation of INT where CAPEST instructions and data can be inserted. These packets trigger the statistical analysis, wait for the result and help disseminate the information throughout the network. An example can be seen in Figure 3.3, where empty probe packets injected at the "A" switch trigger the process at the "B" switch and carry its result to "D". CAPEST delegates issues of measurement scheduling and forwarding policies to the network management system. Thus, concerns about probe packet generation and routing are considered outside the scope of this paper.

Figure 3.3: CAPEST's probe packets trigger the measurement procedure and carry the information along.

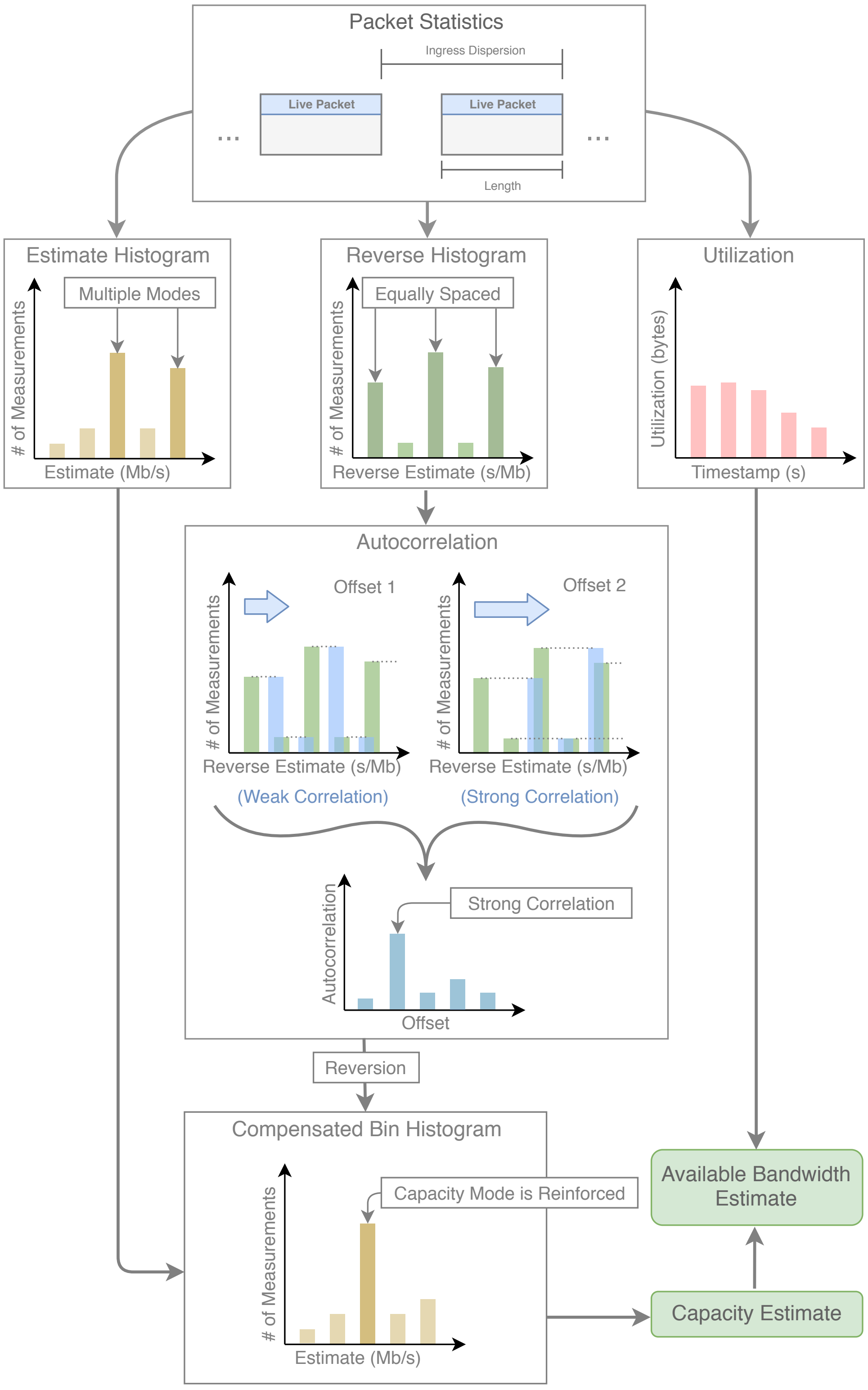


Source: The Author.

3.4 Measurement Method

The CAPEST measurement method determines how the collected statistics are processed to output an estimation of capacity and available bandwidth. The method makes use of dispersion techniques studied by Dovrolis and observations made by Ohkawa and Katti to develop a low complexity capacity estimation method (DOVROLIS; RAMANATHAN; MOORE, 2004; OHKAWA; NOMURA, 2014; KATTI et al., 2004). This process can be divided into three main procedures. The first procedure is the generation of a bin histogram which enables the statistical modal analysis of packet dispersion. The second procedure helps to alleviate common disruptions found in the histogram and consequently reinforces the result, as explained next. The combination of these two functions outputs a decently accurate estimation of capacity, even when operating under reduced processing parameters, most importantly the histogram resolution. The third procedure is an estimate of utilization based on the timestamps and lengths of the last transiting packets. The available bandwidth of the link is derived from utilization and capacity according to Equation 3.2.

Figure 3.4: CAPEST combines an Estimate Histogram with the autocorrelation from a Reverse Estimate Histogram in order to obtain a reinforced capacity measurement. This estimate is subsequently integrated with utilization information to obtain the available bandwidth measurement.



Source: The Author.

3.4.1 Histograms

One of the most prevalent approaches to measuring capacity relies on the analysis of packet dispersion, which is defined as the temporal difference between the complete transmission of two consecutive packets. In the absence of disrupting events, the dispersion δ of two packets sent back-to-back follows Equation 3.5, where l is the length of the packets and C is the effective capacity of the path (DOVROLIS; RAMANATHAN; MOORE, 2004). These principles of packet dispersion have been applied both to end-to-end active methods (DOVROLIS; RAMANATHAN; MOORE, 2004) as well as passive methods (KAPOOR et al., 2004)(OHKAWA; NOMURA, 2014).

$$\delta = \frac{l}{C} \tag{3.5}$$

A common disruption of this estimate is caused by cross-traffic packets interfering in the queuing delay. Since both packets can encounter this additional delay, the dispersion measurement can be distorted upwards or downwards. To mitigate this issue, multiple dispersion samples are collected and statistically analyzed.

CAPEST achieves the aforementioned analysis by constructing bin histograms from packet statistics, as can be seen in Figure 3.4. Bin histograms quantify the occurrences of estimates within a set of intervals, also known as bins. This step, by itself, can yield an accurate estimation of capacity as a global mode. However, the Capacity Mode (CM) can be overshadowed due to the disturbances and often appears only as a local mode.

3.4.2 Autocorrelation

Considering that there are several factors which can obfuscate the capacity estimation taken directly from the bin histogram, such as direct and reverse cross-traffic interference. Identifying patterns in this interference allows the development of statistical procedures to mitigate disturbances. One such observation is that the estimate histogram tends to present local modes at integer fractions of the CM, as can be seen in the multiple modes of the estimate histogram in Figure 3.4. This can be attributed to the fact that cross-traffic interference manifests itself discretely, *i.e.* according to the incidence of cross-traffic packets. This has been previously observed and utilized by related works such as Ohkawa and Katti (OHKAWA; NOMURA, 2014) (KATTI et al., 2004). Considering that queueing delay depends on capacity under transmission time, local modes found at integer fractions of the capacity represent combinations of packets interfering with dispersion.

In order to compensate for this effect, CAPEST constructs a reverse estimate bin histogram, as shown in Figure 3.4. Whereas the regular histogram presents local modes at fractions of the CM, the reverse estimate histogram presents these modes at equally spaced intervals. As an example, a bin histogram presenting modes at $C/1$ Mb/s, $C/2$ Mb/s, $C/3$ Mb/s would generate a reverse bin histogram with equally spaced modes at $1/C$ s/Mb, $2/C$ s/Mb, $3/C$ s/Mb. Subsequently, autocorrelation is applied in order to determine this spacing. This procedure evaluates the correlation of a signal with itself at differing levels of offset. Thus, consistently spaced modes contribute significantly at the same offset, producing a spike. Lastly, this result is reversed back and combined with the estimate histogram by multiplying their contributions at respective bin intervals. This step is depicted as "Reversion" in Figure 3.4 and helps to reinforce the CM.

3.4.3 Utilization

CAPEST collects the timestamps and lengths of packets in order to estimate the recent utilization, as shown on the right side of Figure 3.4. The average utilization U can be calculated from the accumulated length of packets l withing a time interval Δt, following Equation 3.6.

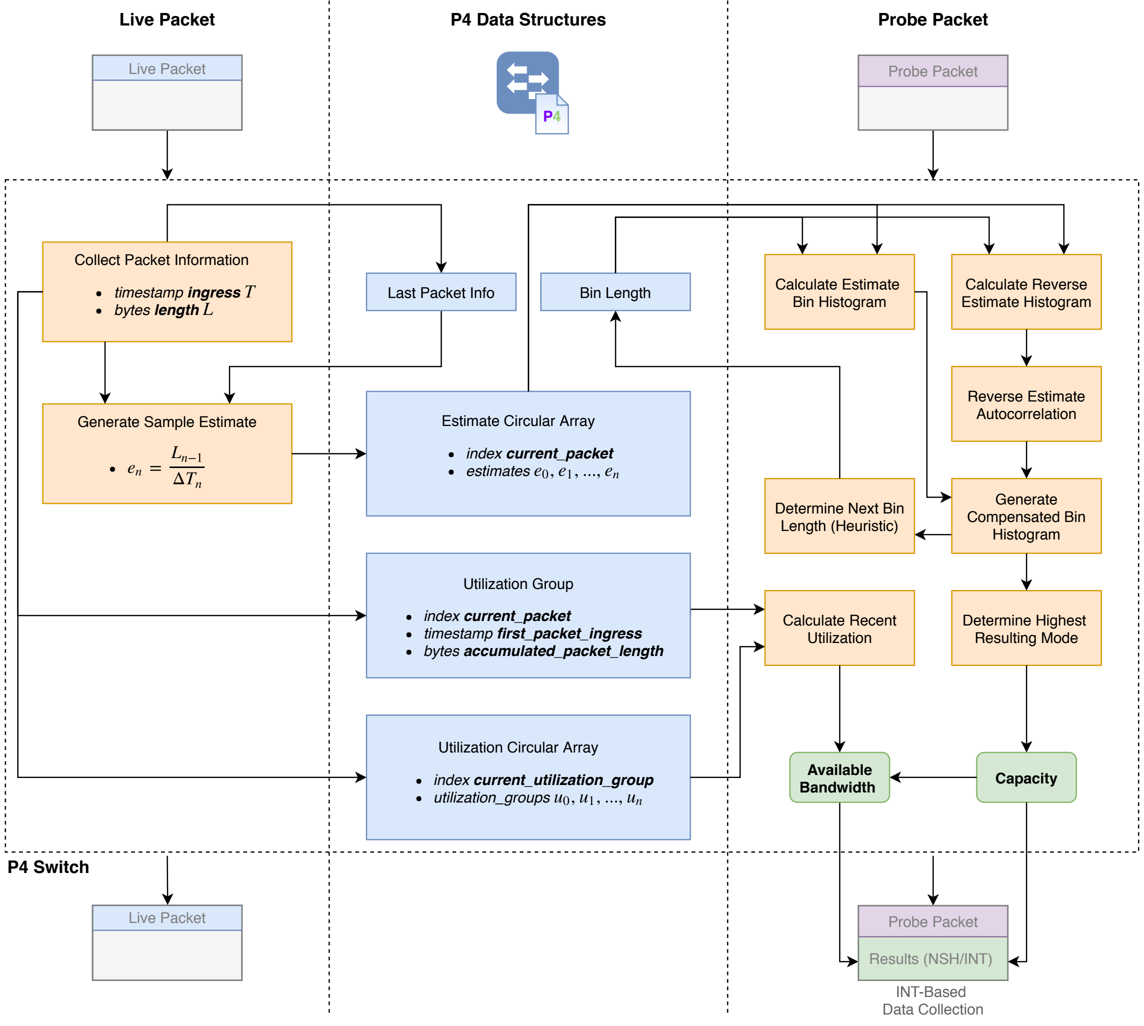
$$U = \frac{\Sigma l}{\Delta t} \tag{3.6}$$

The average utilization is subsequently combined with the estimate of capacity, resulting in an estimate of available bandwidth.

4 CAPACITY ESTIMATION IN P4

In Chapter 4, we elaborate on how the CAPEST method is implemented in the framework of data plane programmability. More specifically, we approach how CAPEST makes use of algorithm parametrization, data structures, packet triggers and how it overcomes limitations contextual to P4.

Figure 4.1: The flow of information between CAPEST structures, procedures and domains.


Live Packet
P4 Data Structures
Probe Packet
Live Packet
P4
Probe Packet
Collect Packet Information
timestamp ingress T
bytes length L
Last Packet Info
Bin Length
Calculate Estimate Bin Histogram
Calculate Reverse Estimate Histogram
Reverse Estimate Autocorrelation
Generate Sample Estimate
$e_n = \frac{L_{n-1}}{\Delta T_n}$
Estimate Circular Array
index current_packet
estimates $e_0, e_1, ..., e_n$
Determine Next Bin Length (Heuristic)
Generate Compensated Bin Histogram
Utilization Group
index current_packet
timestamp first_packet_ingress
bytes accumulated_packet_length
Calculate Recent Utilization
Determine Highest Resulting Mode
Utilization Circular Array
index current_utilization_group
utilization_groups $u_0, u_1, ..., u_n$
Available Bandwidth
Capacity
P4 Switch
Live Packet
Probe Packet
Results (NSH/INT)
INT-Based Data Collection

Source: The Author.

4.1 Live Packet Procedures

Whenever a production packet ingresses at a switch, a few extraction procedures are triggered. As illustrated by the left section of Figure 4.1, a function collects relevant information from the packet, namely the ingress timestamp and the packet length. This data is promptly used by the following procedure and is later used to update both the utilization estimation structures as well as a register with information about the last packet. The second procedure is the generation of a sample estimate as given by the Equation 4.1, where T_{n-1} and L_{n-1} are the last packet's ingress timestamp and length, and T_n is the current packet ingress timestamp. In order for this equation to be implemented, the P4 switch needs to include division (this requirement and its satisfaction are discussed later in Chapter 6). Afterwards, this sample estimate is stored in a dedicated circular array, which is described in detail in the next section.

$$e_n = \frac{L_{n-1}}{(T_n - T_{n-1})} \tag{4.1}$$

Since the live packet functions merely extract information, the packet is not required to wait for them to finish. This represents another effort to ensure live packets suffer as low intrusiveness as possible (see Figure 4.1).

4.2 Data Structures

This section focuses on the data structures used by CAPEST, describing their purpose and their interaction with the other elements. These structures are depicted in the middle portion of Figure 4.1.

4.2.1 Estimate Circular Array

The estimate generated by each live packet is stored in a circular array. This is implemented by means of a register array and an index, which points to the oldest estimate. Whenever a new live packet arrives, the new estimate is placed at the indexed register array position and the index is incremented by one. If it reaches the size of the array, the index is set to zero. Not only does this implementation minimize writes (requiring no structural rearranging), it also guarantees that all the values in the array are valid, simplifying the read procedures as well. The size of the array is equivalent to the number of packets that are input to the algorithm and represents one of the several parameters that can be optimally tuned for different situations.

4.2.2 Utilization Group Circular Array

In order to save utilization data, CAPEST employs a circular array similar to the one describe above. However, this metric requires the storage of both length and time. Additionally, the number of packets necessary for a decent and stable utilization estimation is usually larger than for capacity estimation under typical operation. In light of these observations, two extra data structures were introduced, namely utilization group and utilization circular array.

The utilization circular array stores length and timestamps, but, instead of packet-level information, its data regards groups of packets. The utilization group stores the ingress timestamp of its first packet, the number of packets it has incorporated and their accumulated length. Whenever a live packet crosses the switch, its information is inserted into the utilization group. When the group reaches a certain number of packets, the summarized information of length and time is stored in the circular array and the group is reset. This approach presents two parameters, the number of groups and the number of packets per group. An increase in the latter produces both a reduction in memory utilization as well as an increase in the interval between consecutive updates. In turn, a lower number of packets per group could benefit convergence at the cost of memory utilization (an exhaustive analysis of these parameters is left as future work).

4.3 Probe Packet Procedures

This section relates to the measurement procedures triggered by the probe packet and is displayed in the right part of Figure 4.1.

4.3.1 Histogram Generation

The histogram represents the statistical foundation upon which CAPEST methods operate. Both the estimate histogram as well as the reverse estimate histogram result from an iteration over the estimate circular array. In order to quantify the estimate bins, each sample is divided by the bin length and attributed to a saturated bin index. The reverse estimate methodology inverts the samples and the bin length. At this stage, the float construct would typically be used to conveniently represent samples and their inverted values without losing accuracy. However, there is currently no support for float in P4 (version 16). We considered that implementing floating point functionality would be onerous and bring little benefit considering the flexibility of the histogram system. In light of this reasoning, inverted samples are converted into a different base unit to keep resolution.

4.3.2 Autocorrelation

In the autocorrelation procedure, the reverse estimate histogram is repeatedly correlated with an offset version of itself. At first, the histogram mean and variance are calculated, which are required for correlation. Then, the correlation of each bin and its offset bin is calculated for each offset value between zero and the number of bins.

This procedure entails the use of a dynamic loop nested in a static loop. Considering that P4 does not provide loop primitives, static loops need to be unrolled into a sequence of instructions. Additionally, since P4 methods require a constant number of instructions, dynamic loops must be made static, which can be achieved with conditional statements. In this context, C-Style macros were adopted to improve code readability and parametrization. The insights into implementing this control structure in P4 are further discussed in Chapter 6. Because of the quadratic nature of nested loops, the number of bins represents a very influential parameter, both to performance requirements and statistical granularity. As such, a more in-depth evaluation of its impact is presented in Chapter 5.

4.3.3 Compensated Bin Histogram

After the autocorrelation has taken place, its output needs to be combined with the estimate histogram in order to produce the final capacity estimation. For this to happen, the autocorrelation indices need to be mapped to the original histogram. In this procedure, for each bin index f_n in the frequency histogram, the corresponding autocorrelation index (also known as temporal index t_n) is determined according to Equation 4.2.

$$t_n = \frac{conversion\_pivot}{f_n \times bin\_length} \times \frac{1}{\#t\_bins} \tag{4.2}$$

This translation is necessary because all bin ranges are equally spaced, and thus the estimate ranges do not match one-to-one with the reverse ranges. In order to keep resolution through unit conversion, a relatively large integer (also known as a pivot) is applied. After this step, the bin value with the highest product of these factors determines the capacity estimation.

4.3.4 Bin Length Heuristic

The bin length represents the range of estimates attributed to each bin, effectively determining the granularity of estimation. Thus, in order to keep an accurate and scalable estimation given a limited number of bins, the bin length is dynamically updated according to packet dispersion. In conventional implementations of histogram-driven capacity estimation, the bin length is determined as a percentage of the inter-quartile length of packet estimates (DOVROLIS; RAMANATHAN; MOORE, 2004). However, this would demand a considerably onerous addition to the algorithm, as it would require the packet estimates to be sorted. Time complexity would increase to from $\mathcal{O}(n)$ to $\mathcal{O}(n \log n)$ with respect to number of packets, while also burdening the already restricted switch memory. To avoid such an unaffordable overhead, our proposed mechanism adjusts the bin length – at each probe trigger – according to the histogram count and variance. Three distinct cases guide this adjustment, as shown in Algorithm 1 and described next.

- If the last bin has the highest count, the bin length increases slightly and proportionally (by $increase\_factor$) as it may indicate that most estimates are saturated because they are above the current representative range.
- Else, if there is too much variance, it is decreased proportionally (by $decrease\_factor$). A high variance indicates that counts are poorly distributed, which occurs when a large bin length classifies most counts together. If this happens on a low bin, it hinders accuracy, as the resolution capability is being wasted on higher bins.
- At last, by the same principle explained above, if the first bin contains a large number of estimates (*e.g.*, a $zeroth\_bin\_limit$ of 80%), the bin length is halved to accelerate convergence.

Algorithm 1 Bin Length Heuristic

```
1: if $capacity\_estimate > (\#_{bins} - 1) \times bin\_size$
        then $bin\_size \leftarrow bin\_size + \frac{bin\_size}{increase\_factor}$
2: else if $(capacity\_estimate < (\#_{bins} - 2) \times bin\_size) \&\& (variance > (var\_limit))$
        then $bin\_size \leftarrow bin\_size - \frac{bin\_size}{decrease\_factor}$
3: if $bins[0] > zeroth\_bin\_limit \times \#_{packets}$
        then $bin\_size \leftarrow \frac{bin\_size}{2}$
```

This procedures presents an asymmetry in which it decreases the bin length more efficiently than it increases it. The ideal bin length is the smallest amount capable of representing the CM, since bin length is inversely proportional to accuracy. This effect locates the CM preferably at the higher bins, allowing the heuristic to be less conservative with lower bins and, thus, more effective at decreasing estimations. The heuristic presents a number of parameters which ultimately influence the accuracy and convergence time of CAPEST. In Chapter 5, we explore this relationship in finer detail.

4.3.5 Recent Utilization

Finally, the available bandwidth estimation requires the assessment of recent utilization. This information is calculated by the sum of lengths of all entries l_i in the Utilization Circular Array divided by the time elapsed since the oldest entry t_o, as given by Equation 4.3. The circular array dispenses with control structures, allowing this procedure to be more efficient.

$$U = \frac{\sum_{i=0}^{G} l_i}{(t - t_o)} \tag{4.3}$$

To summarize, in this Chapter we displayed how the algorithm was adapted to the P4 environment. We detailed how to deal with dynamic and static loops in the context of a language with constant number of instructions and without loop primitives. Additionally, instead of implementing the traditional bin length calculation (which would be prohibitive on account of its time complexity), we developed a heuristic that conserves accuracy using the arithmetic and conditional constructs available in P4. We further discuss these aspects in Chapter 6.

5 EXPERIMENTAL SETUP

In this chapter, we evaluate the conceptual and technical feasibility of the proposed mechanism. We start by describing the experimental methodology, in Section 5.1, elaborating on which systems, programs and methods were employed in the assessment. We then present, in Section 5.2, a study on how CAPEST behaves under different parameters and conditions. In Section 5.3, we compare the proposed method to state-of-the-art approaches, focusing on the aspects of freshness and overhead. Finally, we demonstrate a real-world scenario, giving insight into how providers can employ CAPEST to capitalize on video routing in Section 5.4.

5.1 Experimental Methodology and Setup

In order to simulate the behavior of a P4 switch, the CAPEST experimental setup employed the behavioral model p4c-bmv2 version 1.7.0-8f4abeaa[1] compiling for a simple-switch architecture modified to include division as an arithmetic operation[2]. Network emulation was achieved with Mininet version 2.3.0d1, providing integration with virtual hosts and network link configuration. The network topology employed was composed of four P4-enabled switches connected linearly and attached to a host each. Although the topology design is simple, it is capable of capturing CAPEST's main characteristics and allows very controllable scenarios. These emulation tools were adapted from the official P4 language repositories and represent an authentic functional rendition of the P4 environment. Different capacities were dynamically attributed to the the links using Traffic Control (TC). The experiments were executed on a i7-7700 CPU 3.60 GHz 16 GB RAM machine running Ubuntu 14.04.5 Trusty with kernel version 3.13.0-24-generic x86_64. The measured traffic ran through a path with CAPEST-enabled switches and was generated using Iperf version 3.0.7, in line with benefits previously explored by Sommers *et al.* (SOMMERS; BARFORD; WILLINGER, 2006). We artificially inserted standard ICMP packets in order to generate probe packets and extract metrics such as latency, jitter and loss from the path taken by both probe and live packets.

[1]The P4 behavioral model, compiler and simple-switch can be found on the P4 language github page at https://github.com/p4lang.

[2]The complete set of parameters, CAPEST's $P4_{16}$ source code and the scripts utilized in the experiments are available online at https://github.com/nicolaskagami/capest.

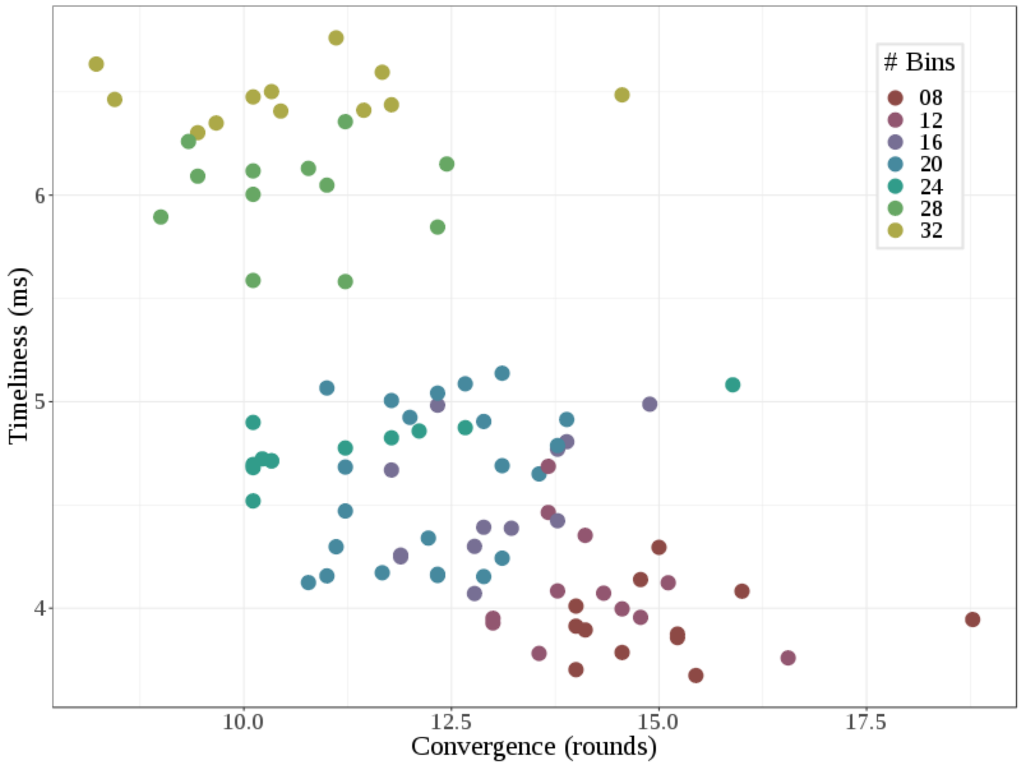
5.2 Sensitivity Analysis

The main objective of this assessment is to determine how the key CAPEST parameters and conditions influence its behavior and performance. As will be presented henceforth, this analysis gives insight into trade-offs and sheds light on how CAPEST can be properly tuned to different scenarios.

5.2.1 Number of Bins Factorization

As previously discussed in Section 3.4, the histogram-based capacity estimation method depends both on the bin length, *i.e.*, the range associated with each bin, as well as the number of bins available to classify each estimate. The bin length is variable and is adjusted at each execution. On the other hand, the number of bins B_n must remain fixed in order to maintain the constant number of instructions as required by P4. This factor is defined at compile time and does not need to be dynamic to yield accurate measurements.

Figure 5.1: The number of bins (color-coded) mapped according to timeliness and convergence.



Source: The Author.

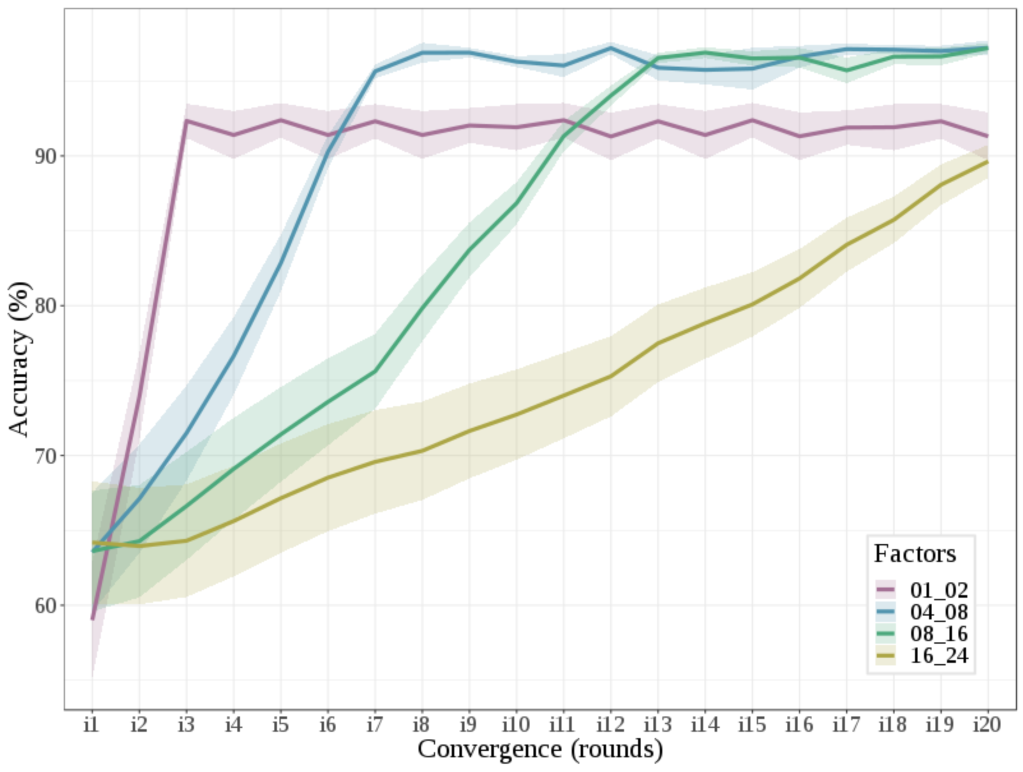
Figure 5.1 illustrates the average convergence[3] and timeliness measured for CAPEST with different numbers of bins. As can be observed, increasing B_n helps to decrease the number of rounds required to converge. This can be explained by the fact that each additional bin increases the instantaneous range of estimation without decreasing the resolution. Consequently, the representation effectively reaches the CM faster without detriment to accuracy. Another important observation is that B_n affects the timeliness of the process in the switch, as illustrated in Figure 5.1. This aspect of timeliness is proportional to the number of instructions executed in the packet method, which increases with $(B_n)^2$. Considering that the advantage observed in convergence time can encounter diminishing returns and that the cost in timeliness increases exponentially, a sweet spot can be identified for a given scenario.

5.2.2 Bin Length Heuristic Factorization

The bin length heuristic is strongly influenced by two factors, which determine the upwards and downwards convergence. These are inversely proportional to the current bin length, *i.e.*, a factor of “1” represents an adjustment of 100% of the bin length while a factor of “4” represents 25%. With this in mind, a few observations can be made regarding the desirable relationship between these two factors. If both factors are relatively high (such as the case of the red curve (01_02) in Figure 5.2), the adaptation may be prone to low accuracy due to poor resolution. Conversely, if both factors are relatively low (*e.g.*, close to the number of bins), the number of rounds required to converge increases considerably, as illustrated by the yellow curve (16_24) in Figure 5.2. Considering that the heuristic already presents characteristics favorable to decreasing, as mentioned in Subsection 4.3.4, it can be argued that its increase factor should be relatively higher than its decrease factor.

[3]Convergence was defined as taking place when the no more than 10% measurement variation is observed for 10 consecutive rounds.

Figure 5.2: Average accuracy of the different bin length heuristic factors (color-coded) as they converge.

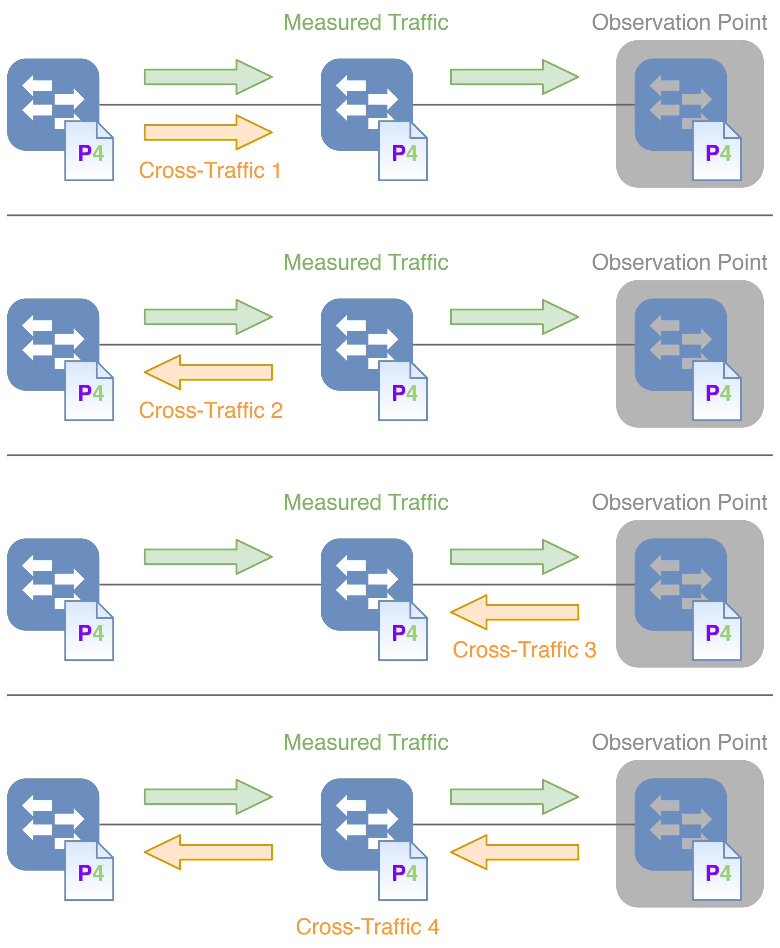


Source: The Author.

5.2.3 Cross-Traffic Sensitivity

Cross traffic represents one of the most important obstacles to be overcome by dispersion-based measurement methods. Thus, it is essential to have insight into how a method performs under strenuous conditions. CAPEST was subjected to different cross-traffic scenarios, varying in three different aspects, as shown in Figure 5.3. The first aspect refers to direction, concerning whether the extraneous traffic is injected in the same or opposite direction to that of the measured traffic. As for the second aspect, the cross traffic can traverse through the local hop (such as CTs 3 and 4) where the Observation Point is located or through a non-local hop shared by the measured traffic. Finally, there is the aspect of persistence, *i.e.*, the amount of a path common to both types of traffic. In this context, path-persistent cross-traffic is defined as traffic that shares the path of measurement.

Figure 5.3: The cross traffic scenarios relative to the measured traffic and the Observation Point.



Source: The Author.

The cross-traffic experiment setup was Full Factorial with four kinds of cross-traffic, five levels of cross-traffic intensity (20%, 40%, 60%, 80%, 100% relative to measurement traffic) and 72 repetitions. All findings are statistically significant considering a confidence level of 95%.

In Table 5.1, we present the capacity accuracy of CAPEST under different cross-traffic conditions, compared to the baseline (without cross traffic). It can be observed that the most influential cross-traffic characteristic is locality regarding the measurement hop. Cross-traffic located at a local hop was found to be nearly twice as impactful as a similar load on a non-local hop, decreasing accuracy by 6.81% on average. Overall, the presence of cross-traffic decreased accuracy by 5.11% on average, leaving the method with close to 90% accuracy on all cases, as is customarily reported by state-of-the-art techniques under normal conditions.

Table 5.1: Cross Traffic Factors and Influence on Accuracy

Case	Direction	Hop	Persistence	Accuracy
Baseline				96.83%
1	Same	Non-local	Hop	93.03%
2	Opposite	Non-local	Hop	93.79%
3	Opposite	Local	Hop	88.08%
4	Opposite	Local	Path	91.96%

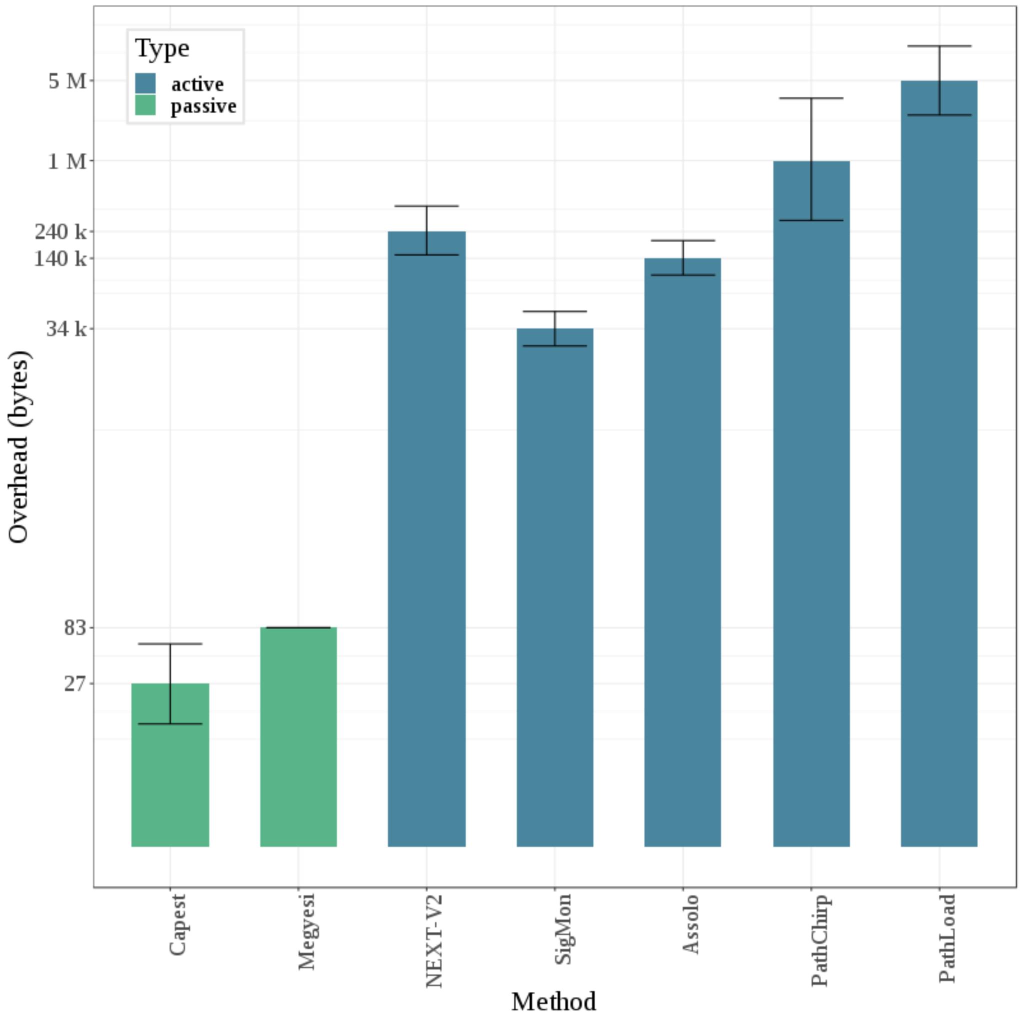
Source: The Author.

5.3 Comparative Evaluation - intrusiveness, Freshness And Deployability

In this section, we aim to compare the proposed method with established network measurement techniques in regards to intrusiveness, freshness and deployability. It should be noted that these techniques diverge in a number of different characteristics, as depicted in Table 2.1. These techniques vary in scope (*i.e.*, path or hop), method type (*i.e.*, passive or active) and most only attempt to measure one of the two target metrics estimated by CAPEST. Moreover, some ABW estimation techniques require prior knowledge of capacity, essentially measuring utilization (as defined in Equation 3.2). In light of this diversity and intending to contrast CAPEST with as many established techniques as possible, the data presented in this section resulted from a comprehensive investigation into self-reported data of the respective academic publications (MEGYESI et al., 2017; PAUL; TACHIBANA; HASEGAWA, 2016; KIM; LEE, 2014; GOLDONI; ROSSI; TORELLI, 2009; KAPOOR et al., 2004; RIBEIRO et al., 2003) and third-party simulations (SALCEDO; GUERRERO; GUÉRRERO, 2017; GUERRERO; LABRADOR, 2010).

Aiming to compare the level of intrusiveness produced by each of the methods, we focus on overhead. Overhead is defined as the amount of data introduced to the network in order to perform the measurement. In this context, active techniques contribute most significantly as they require the injection of packets into the data stream. However, the portion of overhead incurred by passive techniques should not be disregarded. The trace data required by most passive capacity and ABW estimators needs to be sent (at least through the control plane). This requires raw or at least pre-processed data to be transmitted to the collector unit, which is frequently centralized. Moreover, the amount of pre-processing achievable on packet dispersion techniques is limited. For example, the dynamic bin length requires the storage of individual dispersion instances, disallowing summarizing techniques such as sketching (ALON; MATIAS; SZEGEDY, 1999).

Figure 5.4: Logarithmic levels of overhead reported by different capacity and/or ABW estimation techniques.

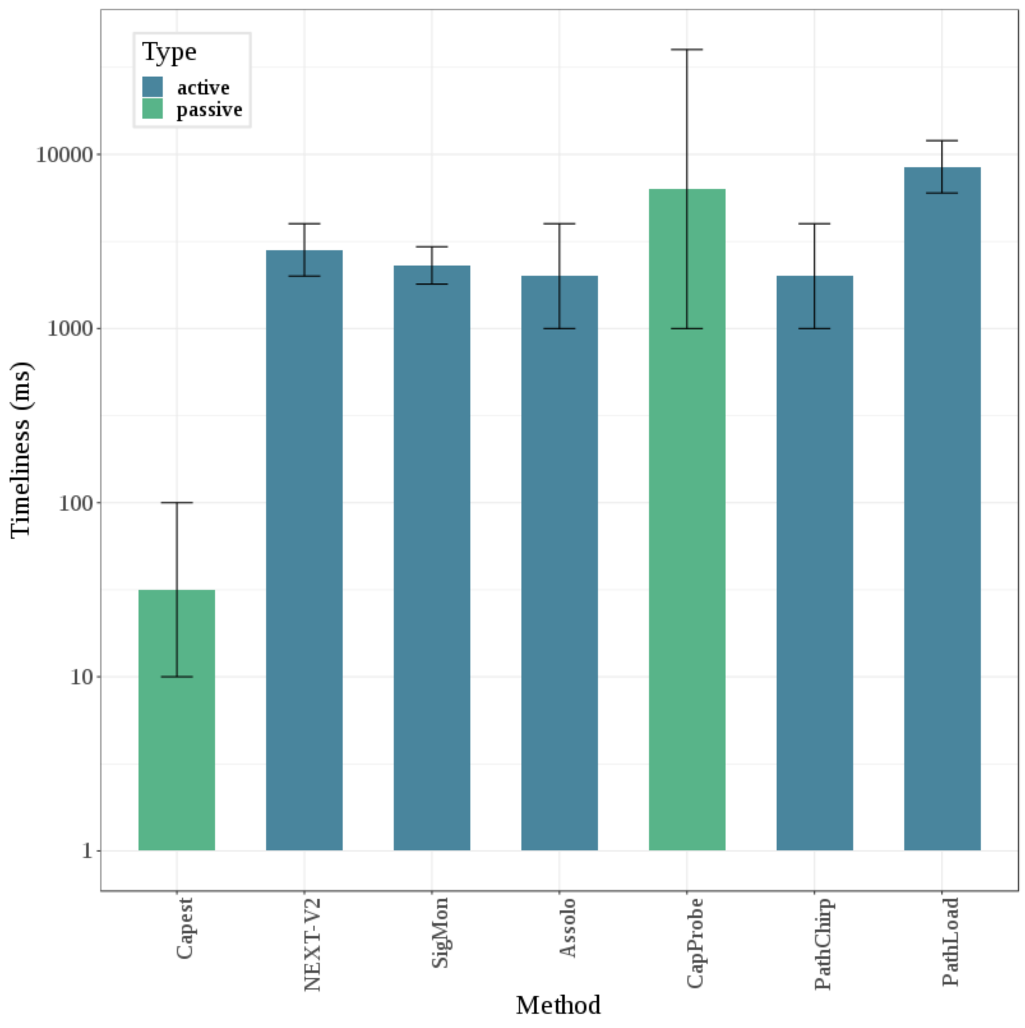


Source: The Author.

Figure 5.4 presents the amount of overhead reported by the comprehended methods on a logarithmic scale. This context clearly illustrates the divide between active and passive methods, conveying a gap of at least two orders of magnitude (100x). The gap between CAPEST and the next least intrusive method is also significant, especially considering that the Megyesi method presumes to know capacity and only measures utilization. CAPEST achieves lower overhead than other passive methods because it doesn't require the transfer of input data. The estimation is accomplished *in-situ*, sending only the result through the network, which amounts to 12 bytes per port per round.

Among the significant contributing factors to the freshness of a measurement method are the availability of fresh data and the timeliness of the process. In this context, active techniques have the advantage of providing their own packets, which guaran-

Figure 5.5: Logarithmic degrees of timeliness reported by the considered techniques.



Source: The Author.

tees data freshness. However, this cuts both ways, as these techniques need time to insert enough data to provide statistical significance comparable to having access to the network history. In turn, the freshness of data used by any passive technique depends on the underlying traffic. However, measurement timeliness can be improved with a faster processing and delivery of data. In the local scope of the switch, the result of CAPEST is available as soon as the processing takes place, which happens at line rate. Since our method can request a measurement and receive its result in a round-trip's time, it offers the best timeliness reported by any of the studied techniques by a factor of 10, as demonstrated in Figure 5.5.

Regarding deployability, the gathering of data traces from the network imposes an obstacle for most passive methods. These solutions need to either be constantly gathering information, which continuously consumes resources, or actively trigger the collection, which requires time and hinders freshness. CAPEST's *in-situ* processing negates this drawback, reducing resource utilization and simplifying the estimation procedure.

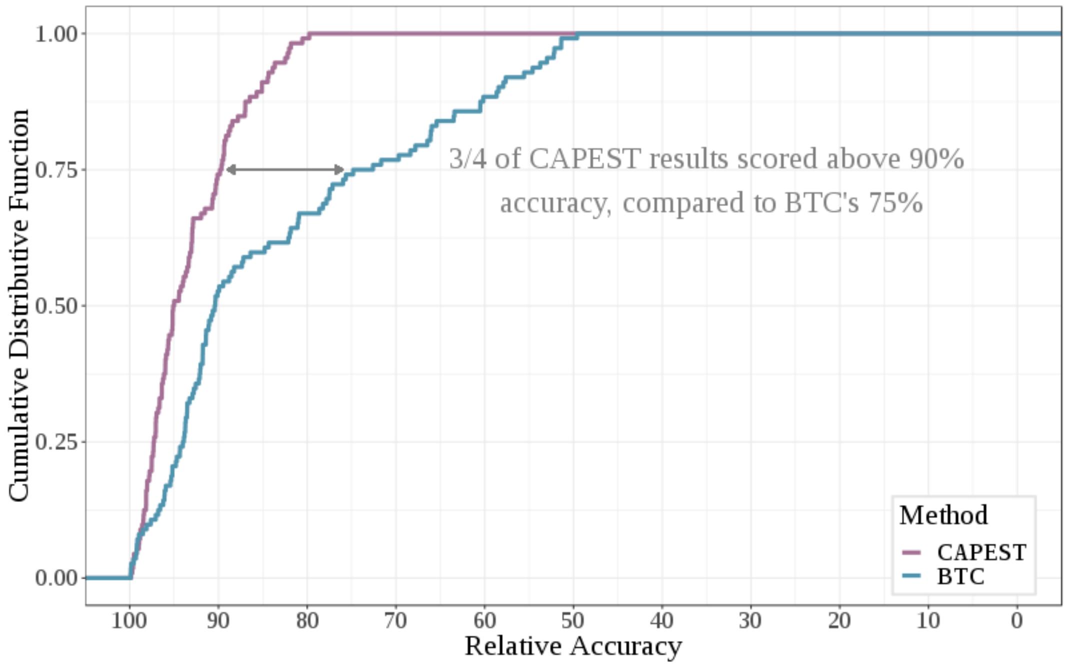
5.4 Real World Scenario - Video Streaming

In this section, we showcase one of the possible applications of CAPEST for measuring available bandwidth in the case of video streaming, highlighting advantages to both accuracy and reduction of intrusiveness (*i.e.*, the amount of degradation of network services introduced by the measurement method). Video streaming currently comprises a significant portion of all network traffic. Mobile video, specifically, is expected to constitute 78% of all data traffic in 2021 (CISCO, 2017). As an effort to balance infrastructure optimization and user Quality-of-Experience (QoE), network providers can resort to traffic engineering such as QoE-aware path selection algorithms (FILHO et al., 2018). In this context, it is essential to acquire an accurate ABW estimation, as it represents the single most influential Quality-of-Service predictor of QoE (CASAS et al., 2016; RAMAKRISHNAN et al., 2015). In order to guarantee decent accuracy and freshness in this measurement, active methods such as Bulk-Transfer Capacity (BTC) estimators are typically employed (FILHO et al., 2018). This process produces two complications: (*i*) the heavy amount of overhead introduced by the method can degrade QoE, as the very target of measurement is being consumed; and (*ii*) accuracy is not perfect since BTC is not exactly the same as ABW (JAIN; DOVROLIS, 2004) and the sporadic nature of video buffering can compromise the measurement.

In the subject of streaming, one of the strongest indications of performance degradation (and its consequent depreciation of QoE) is video stall (NAM; KIM; SCHULZRINNE, 2016; CASAS et al., 2016), which occurs when playout is interrupted. In order to faithfully reproduce the video playout ecosystem while evaluating ABW accuracy and intrusiveness, we employed a video client simulator capable of recording stalls (FILHO et al., 2018). The client behaviour is analogous to that of a Youtube playout, with a sixty-second buffer constantly being filled with ten-second chunks which are consumed in real time. Whenever the buffer is depleted, a stall event and its length are registered.

This experiment consisted of multiple concomitant videos passing through the observation point of CAPEST. The number of videos and capacity varied in order to consider multiple levels of ABW. These levels ranged from 85.46% of capacity to a case where total video bandwidth demand is higher than capacity, incurring in QoE degradation. CAPEST's ABW estimation is contrasted with a BTC estimator represented by Iperf. After each iteration, the estimate is compared with the nominal capacity minus the aggregated bitrate and the average stall length is accounted.

Figure 5.6: Cumulative relative accuracy of CAPEST and BTC.

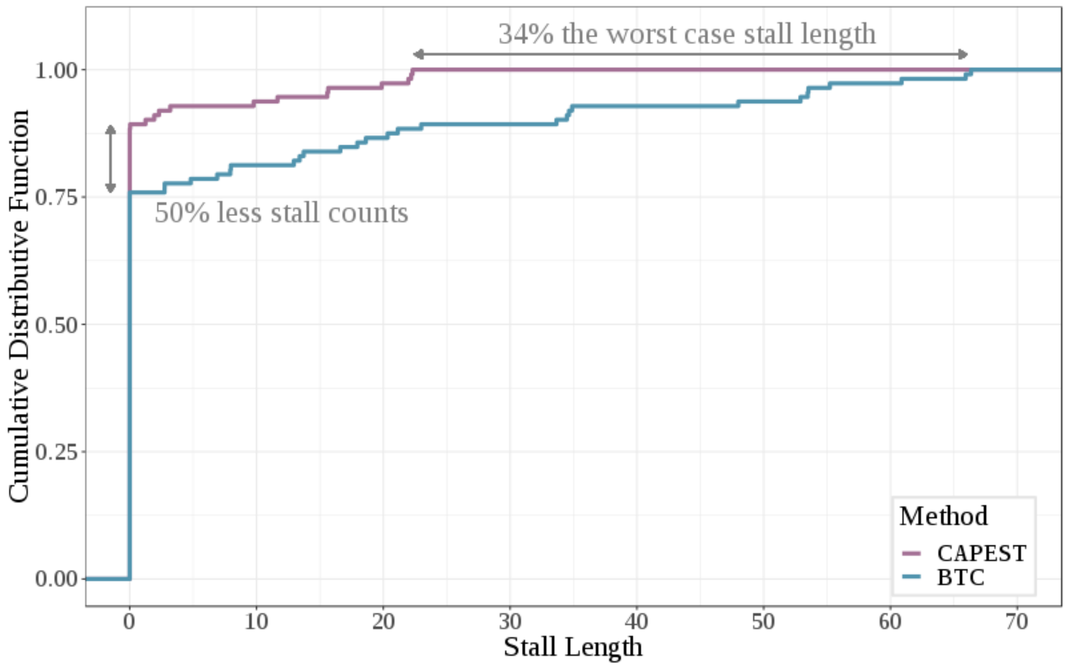


Source: The Author.

Figure 5.6 illustrates the obtained results, in which CAPEST clearly outperforms BTC regarding accuracy. Three quarters of CAPEST estimates achieved 90%+ accuracy, while BTC achieves the same accuracy for only 75% of the cases. This can be partially attributed to the fact that BTC is not equal to ABW. It, by definition, interacts with the current traffic via congestion control. Since the video buffering rate is intermittent (*i.e.*, it loads chunks periodically), the BTC estimator risks not interacting with all currently active streams proportionally during its measurement. In order to alleviate this complication, the measurement duration is increased. It should be greater than the time taken to consume each chunk[4] to increase the chance of interacting proportionally with all streams. However, this practice also increases intrusiveness.

[4]The measurement duration used for BTC in this experiment was 60s, which is greater than the chunk consumption time of 10s.

Figure 5.7: Cumulative average stall length per video of CAPEST and BTC.



Source: The Author.

In the context of intrusiveness, Figure 5.7 depicts the cumulative distribution function of the average observed stall length per video of CAPEST and BTC. The use of CAPEST for video ABW estimation achieved a reduction of 81.42% in stall length, compared to measuring with BTC. BTC produced stall in 2 times as many scenarios as CAPEST and its worst stall was 2.98 times as long. While BTC needs to inject a substantial amount of traffic, CAPEST only requires the estimate to be triggered and sent, suggesting a much better fit to the case of video traffic ABW estimation.

6 DISCUSSION

As previously mentioned, P4 provides a relatively restricted set of tools to operate in the data plane. This restraint is conscious and built-in with the honorable goal of stability and efficiency. However, in the context of a more sophisticated packet method such as CAPEST, a few design approaches had to be developed in order to contour these limitations. This chapter covers the lessons learned during the development of the proposed method.

6.1 Introducing Complex Arithmetic Operations

Complex arithmetic operations can be incorporated. Current P4 implementations do not include multiplication, division or modulo as run-time operations. The reason is twofold: First, such operations are not often needed for most packet processing applications, as initially claimed by the P4 development community. Second, there is a commitment to constant and efficient packet processing in the P4 language specification, which could be hindered by the inclusion of more complex operations considering that they generally require more cycles to complete. While early switch programmability focused on basic operations to mostly realize forwarding procedures, present and future implementations should seek to expand the scope of tasks deployable on the data plane, closely followed by advancements to the switch architecture. In this context, the ability to divide in run-time enables a number of applications (*e.g.* unit conversion, statistical analysis), including our own. With this in mind, we tweaked the existing implementation of the simple switch for CAPEST, replacing an unused basic operation with division. Thus, we conclude that the inclusion of division into the P4 specification would be beneficial, even as an optional element, to encourage the development and study of more diverse data plane applications.

6.2 Dealing with loops

The absence of loops can be remedied. The P4 specification does not establish any loop primitive, in concord with the principle of constant delay. However, both dynamic and static loops are commonly employed by statistical methods. In order to deal with this complication, static loops can be unrolled into a list of operations. The number of operations, which needs to be constant, can represent a parameter of the P4 program. In this context, the C-style macros adopted by P4 allow static loops to be customized through repetition, contributing to cleaner source code. Considering the widespread adoption of loops, the P4 specification could benefit from a dedicated construct for more practical iterators.

Algorithms that rely on dynamic loops can also be implemented if the dynamic loop is converted to a static loop. This conversion entails the creation of a static loop of size equal to the maximum number of iterations, with the addition of a conditional construct to neutralize the excess iterations. Under these conditions, the time complexity of an algorithm becomes constant and equal to its worst-case scenario. These circumstances exemplify how a low-level decision in favor of predictability results in a paradigm shift at the developmental level, influencing the balance between performance and predictability.

6.3 Applying Heuristics

Limited space encourages the use of heuristics. One of the challenges brought about by the space limitations imposed by switches is the use of exact algorithmic procedures. Considering the detriment of reducing operational parameters to accommodate these procedures, it may be more suitable to apply lightweight heuristics. In the context of CAPEST, the bin length is customarily determined by a fraction of the interquartile length of packet estimates (DOVROLIS; RAMANATHAN; MOORE, 2004). This property can only be acquired after these estimates are sorted, which would impose a long and complex procedure. Even at optimal implementations with time complexity of $\mathcal{O}(n \log n)$ (where n is the number of analyzed packets), this would require a considerable increase to the method, since the number of packets is the highest parameter by far. The applied heuristic presents an extremely small footprint as it uses pre-calculated data for the most part, leaving the overall algorithm with complexity $\mathcal{O}(n)$ with respect to number of packets. The trade-off in convergence can be considered negligible in light of the timeliness and overall timescale at which these procedures operate. In our case, the advantage to convergence time provided by the interquartile length was deemed not to be worth the increase in space and time complexity, especially when compared to an equivalent increase in the base parameter (number of packets). Essentially, when working with limited resources, heuristics can be a more cost-effective approach than exact algorithms.

6.4 Improving Resource Utilization

Differentiated packet methods can improve resource utilization. When first confronted with the development of a statistical data plane application, one might be tempted to produce a one-size-fits-all method containing the mechanism in its entirety. However, considering current tools and circumstances, there are more efficient configurations. The statistical process, albeit reduced in complexity and duration in this endeavor, can still incur tangible resource utilization. Additionally, it may be favorable that the information collection and the statistical analysis be disassociated, which is made possible by configurable switch internal memory. In the case of CAPEST, there is a concern that the live packets experience the lowest degradation possible. With this in mind, our proposed design introduces an artificial packet to trigger the heavier estimation procedure, lifting the processing burden. The effectiveness of this approach depends on a few architectural characteristics.

- First, if the line rate is defined by the slowest method, affecting all transiting packets, the separation of methods still reduces utilization, as live packets do not trigger statistical analyses.
- Second, if a dedicated queue (slower than line rate) is available, live packets will only wait for their brief collection procedures to terminate before exiting the switch. The existence of a slow path is suggested in the INT specification as an approach to bulkier packet procedures (Changhoon Kim et al., 2016).
- Third, if a packet is allowed to leave the switch before its method finishes, live packets would experience even less delay. Since the live packet is not modified by its procedure, it would be free to leave after its relevant information is extracted.This approach could be achieved by packet copying mechanisms, also suggested in the INT specification (Changhoon Kim et al., 2016).

In conclusion, the current P4 environment already allows the decoupling of complex methods to reduce resource utilization. However, this approach can be further improved if P4 embraces newly-envisioned architectural features such as dedicated queues and smart copying mechanisms.

7 CONCLUSION

Current network measurement techniques face the predicament of having to compromise on accuracy, intrusiveness or freshness in order to achieve capacity and available bandwidth estimation. In order to address this issue, this dissertation proposed CAPEST, a passive estimation method leveraging data plane programmability to execute a packet dispersion statistical analysis *in-situ*. To the best of our knowledge, it is the first method to measure capacity and ABW from the data plane. The method employs reverse estimate autocorrelation to reinforce the capacity estimation and a heuristic to calculate bin length with linear complexity with respect to the number of packets.

The proposed method underwent a comprehensive evaluation, in which it presented less intrusiveness and an improvement of 10x to freshness compared with state-of-the-art techniques. A sensitivity assessment was performed, illustrating how parametrization affects overall behavior and showing how CAPEST managed to measure capacity at near 90% accuracy in the presence of four different types of cross traffic. CAPEST was subjected to a real-world application of video traffic and measured available bandwidth 10% more accurately than its counterpart while incurring in a reduction of 81.42% observed stall length per video. Finally, we detailed key insights from our exploration of data plane programmability. In this overview, we discussed the challenges and opportunities presented by the P4 environment, concerning the interplay between technical issues and developmental paradigms.

There are several directions to build upon the work presented in this dissertation. The bin length heuristic can be improved by making the adjustment dynamic and relative to the rank of the resultant bin. Furthermore, the method can be tailored to work better depending on the architectural circumstances described in Chapter 6. For example, the impact of the slowest packet method in the line rate can be reduced by distributing a few calculations to live packet methods. Considering the suitability of the method for measuring video traffic, it would be interesting to apply it to a QoE-aware path selector and measure its potential in terms of accuracy and resource utilization. The proposed method could also be combined with packet filtering to allow for the evaluation and diagnostic of differentiated services from the data plane.

REFERENCES

ALON, N.; MATIAS, Y.; SZEGEDY, M. The space complexity of approximating the frequency moments. **Journal of Computer and system sciences**, Elsevier, v. 58, n. 1, p. 137–147, 1999.

ANWER, M. B.; FEAMSTER, N. Building a fast, virtualized data plane with programmable hardware. In: ACM. **Proceedings of the 1st ACM workshop on Virtualized infrastructure systems and architectures**. [S.l.], 2009. p. 1–8.

Atary, A.; Bremler-Barr, A. Efficient round-trip time monitoring in openflow networks. In: **IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications**. [S.l.: s.n.], 2016. p. 1–9.

BOSSHART, P. et al. Forwarding metamorphosis: Fast programmable match-action processing in hardware for sdn. In: ACM. **ACM SIGCOMM Computer Communication Review**. [S.l.], 2013. v. 43, n. 4, p. 99–110.

BOSSHART, P. et al. P4. **ACM SIGCOMM Computer Communication Review**, v. 44, n. 3, p. 87–95, 2014. ISSN 01464833. Available from Internet: <http://dl.acm.org/citation.cfm?doid=2656877.2656890>.

BRESLAU, L. et al. Endpoint admission control: Architectural issues and performance. **ACM SIGCOMM Computer Communication Review**, ACM, v. 30, n. 4, p. 57–69, 2000.

CAESAR, M. et al. Design and implementation of a routing control platform. In: USENIX ASSOCIATION. **Proceedings of the 2Nd Conference on Symposium on Networked Systems Design & Implementation-Volume 2**. [S.l.], 2005. p. 15–28.

CASADO, M. et al. Ethane: Taking control of the enterprise. In: ACM. **ACM SIGCOMM Computer Communication Review**. [S.l.], 2007. v. 37, n. 4, p. 1–12.

CASADO, M. et al. Sane: A protection architecture for enterprise networks. In: **USENIX Security Symposium**. [S.l.: s.n.], 2006. v. 49, p. 50.

CASAS, P. et al. An educated guess on qoe in operational networks through large-scale measurements. In: **Proceedings of the 2016 Workshop on QoE-based Analysis and Management of Data Communication Networks**. New York, NY, USA: ACM, 2016. (Internet-QoE '16), p. 1–6. ISBN 978-1-4503-4425-8.

CAVUSOGLU, B.; ORAL, E. A. Estimation of available bandwidth share by tracking unknown cross-traffic with adaptive extended Kalman filter. **Computer Communications**, Elsevier B.V., v. 47, p. 34–50, 2014. ISSN 01403664. Available from Internet: <http://dx.doi.org/10.1016/j.comcom.2014.04.008>.

Changhoon Kim et al. **In-band Network Telemetry (INT)**. [S.l.], 2016. 1–28 p. Available from Internet: <http://p4.org/wp-content/uploads/fixed/INT/INT-current-spec.pdf>.

CISCO. **Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2016–2021**. [S.l.], 2017.

CORDEIRO, W. L. da C.; MARQUES, J. A.; GASPARY, L. P. Data plane programmability beyond openflow: Opportunities and challenges for network and service operations and management. **Journal of Network and Systems Management**, Springer, v. 25, n. 4, p. 784–818, 2017.

DOVROLIS, C.; RAMANATHAN, P.; MOORE, D. What do packet dispersion techniques measure? In: **Proceedings IEEE INFOCOM 2001. Conference on Computer Communications. Twentieth Annual Joint Conference of the IEEE Computer and Communications Society (Cat. No.01CH37213)**. [S.l.: s.n.], 2001. v. 2, p. 905–914 vol.2. ISSN 0743-166X.

DOVROLIS, C.; RAMANATHAN, P.; MOORE, D. Packet-dispersion techniques and a capacity-estimation methodology. **IEEE/ACM Transactions on Networking**, v. 12, n. 6, p. 963–977, 2004. ISSN 10636692.

DOWNEY, A. B. Using pathchar to estimate internet link characteristics. In: **Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication**. New York, NY, USA: ACM, 1999. (SIGCOMM '99), p. 241–250. ISBN 1-58113-135-6. Available from Internet: <http://doi.acm.org/10.1145/316188.316228>.

FEAMSTER, N.; REXFORD, J.; ZEGURA, E. The road to sdn: an intellectual history of programmable networks. **ACM SIGCOMM Computer Communication Review**, ACM, v. 44, n. 2, p. 87–98, 2014.

FILHO, R. I. T. da C. et al. Network fortune cookie: Using network measurements to predict video streaming performance and qoe. In: **2016 IEEE Global Communications Conference (GLOBECOM)**. [S.l.: s.n.], 2016. p. 1–6.

FILHO, R. I. T. da C. et al. Scalable QoE-aware path selection in SDN-based mobile networks. In: **IEEE INFOCOM 2018 - IEEE Conference on Computer Communications (INFOCOM 2018)**. Honolulu, USA: [s.n.], 2018.

GOLDONI, E.; ROSSI, G.; TORELLI, A. Assolo, a New Method for Available Bandwidth Estimation. **2009 Fourth International Conference on Internet Monitoring and Protection**, p. 130–136, 2009. Available from Internet: <http://ieeexplore.ieee.org/document/5076361/>.

GUERRERO, C. D.; LABRADOR, M. A. On the applicability of available bandwidth estimation techniques and tools. **Computer Communications**, Elsevier B.V., v. 33, n. 1, p. 11–22, 2010. ISSN 01403664. Available from Internet: <http://dx.doi.org/10.1016/j.comcom.2009.08.010>.

JAIN, M.; DOVROLIS, C. Pathload: A measurement tool for end-to-end available bandwidth. In: CITESEER. **In Proceedings of Passive and Active Measurements (PAM) Workshop**. [S.l.], 2002.

JAIN, M.; DOVROLIS, C. Ten Fallacies and Pitfalls on End-to-End Available Bandwidth Estimation. **The 4th ACM SIGCOMM conference on Internet measurement Key: citeulike:1718948**, n. 3, p. 272–277, 2004.

JEYAKUMAR, V.; ALIZADEH, M.; GENG, Y. Millions of Little Minions: Using Packets for Low Latency Network Programming and Visibility. **Sigcomm 2014**, p. 1–2, 2014. ISSN 01464833. Available from Internet: <http://arxiv.org/abs/1405.7143>.

JULURI, P.; TAMARAPALLI, V.; MEDHI, D. Measurement of quality of experience of video-on-demand services: A survey. **IEEE Communications Surveys Tutorials**, v. 18, n. 1, p. 401–418, Firstquarter 2016. ISSN 1553-877X.

KAPOOR, R. et al. CapProbe: A Simple and Accurate Capacity Estimation Technique. **ACM SIGCOMM Computer Communication Review**, v. 34, n. 4, p. 67, 2004. ISSN 01464833. Available from Internet: <http://portal.acm.org/citation.cfm?doid=1030194.1015476>.

KATTA, N. et al. Hula: Scalable load balancing using programmable data planes. In: ACM. **Proceedings of the Symposium on SDN Research**. [S.l.], 2016. p. 10.

KATTI, S. et al. Multiq: Automated detection of multiple bottleneck capacities along a path. In: **Proceedings of the 4th ACM SIGCOMM Conference on Internet Measurement**. New York, NY, USA: ACM, 2004. (IMC '04), p. 245–250. ISBN 1-58113-821-0. Available from Internet: <http://doi.acm.org/10.1145/1028788.1028820>.

KIM, C. et al. In-band network telemetry via programmable dataplanes. In: **ACM SIGCOMM**. [S.l.: s.n.], 2015.

KIM, J. C.; LEE, Y. An end-to-end measurement and monitoring technique for the bottleneck link capacity and its available bandwidth. **Computer Networks**, Elsevier B.V., v. 58, n. 1, p. 158–179, 2014. ISSN 13891286. Available from Internet: <http://dx.doi.org/10.1016/j.comnet.2013.08.028>.

KREUTZ, D. et al. Software-defined networking: A comprehensive survey. **Proceedings of the IEEE**, v. 103, n. 1, p. 14–76, 2015. ISSN 00189219. Available from Internet: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6994333>.

MARTÍN-SACRISTÁN, D. et al. On the way towards fourth-generation mobile: 3gpp lte and lte-advanced. **EURASIP Journal on Wireless Communications and Networking**, v. 2009, n. 1, p. 354089, Aug 2009. ISSN 1687-1499. Available from Internet: <https://doi.org/10.1155/2009/354089>.

MCKEOWN, N. et al. Openflow: enabling innovation in campus networks. **ACM SIGCOMM Computer Communication Review**, ACM, v. 38, n. 2, p. 69–74, 2008.

MEGYESI, P. et al. Challenges and solution for measuring available bandwidth in software defined networks. **Computer Communications**, v. 99, p. 48 – 61, 2017. ISSN 0140-3664. Available from Internet: <http://www.sciencedirect.com/science/article/pii/S014036641630648X>.

MELANDER, B.; BJORKMAN, M.; GUNNINGBERG, P. A new end-to-end probing and analysis method for estimating bandwidth bottlenecks. In: **Global Telecommunications Conference, 2000. GLOBECOM '00. IEEE**. [S.l.: s.n.], 2000. v. 1, p. 415–420 vol.1.

MICHAUT, F.; LEPAGE, F. Application Oriented Network Metrology: Metrics and Active Measurement Tools. **IEEE Communications Surveys and Tutorials**, p. 2–24, 2005.

MICHELINAKIS, F. et al. Lightweight capacity measurements for mobile networks. **Computer Communications**, Elsevier, v. 84, p. 73–83, 2016.

MORTON, A. **Active and Passive Metrics and Methods (with Hybrid Types In-Between)**. RFC Editor, 2016. RFC 7799. (Request for Comments, 7799). Available from Internet: <https://rfc-editor.org/rfc/rfc7799.txt>.

MOSHREF, M. et al. Trumpet. **Proceedings of the 2016 conference on ACM SIGCOMM 2016 Conference - SIGCOMM '16**, p. 129–143, 2016. Available from Internet: <http://dl.acm.org/citation.cfm?doid=2934872.2934879>.

NAM, H.; KIM, K. H.; SCHULZRINNE, H. Qoe matters more than qos: Why people stop watching cat videos. In: **IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications**. [S.l.: s.n.], 2016. p. 1–9.

OHKAWA, N.; NOMURA, Y. Path capacity estimation by passive measurement for the constant monitoring of every network path. **APNOMS 2014 - 16th Asia-Pacific Network Operations and Management Symposium**, p. 2–7, 2014.

PAKZAD, F.; PORTMANN, M.; HAYWARD, J. Link Capacity Estimation in Wireless Software Defined Networks. **2015 International Telecommunication Networks and Applications Conference (ITNAC)**, p. 208–213, 2015. Available from Internet: <http://ieeexplore.ieee.org/document/7366814/>.

PAUL, A. K.; TACHIBANA, A.; HASEGAWA, T. An enhanced available bandwidth estimation technique for an end-to-end network path. **IEEE Transactions on Network and Service Management**, v. 13, n. 4, p. 768–781, Dec 2016. ISSN 1932-4537.

PROSAD, R. et al. Bandwidth estimation: metrics, measurement techniques, and tools. **IEEE Network**, v. 17, n. 6, p. 27–35, 2003. ISSN 0890-8044.

RAMAKRISHNAN, S. et al. Sdn based qoe optimization for http-based adaptive video streaming. In: **Proceedings of the IEEE International Symposium on Multimedia (ISM)**. [S.l.: s.n.], 2015. p. 120–123.

REKHTER, Y.; LI, T.; HARES, S. **A border gateway protocol 4 (BGP-4)**. [S.l.], 2005.

RIBEIRO, V. et al. pathchirp: Efficient available bandwidth estimation for network paths. **Passive and Active Measurements (PAM)**, 04 2003. ISSN 03676234.

SALCEDO, D.; GUERRERO, C.; GUÉRRERO, J. Overhead in available bandwidth estimation tools: Evaluation and analysis. **International Journal of Communication Networks and Information Security (IJCNIS)**, v. 9, n. 3, 2017.

SALIM, J. et al. **Linux netlink as an ip services protocol**. [S.l.], 2003.

SCHWARTZ, B. et al. Smart packets for active networks. In: IEEE. **Open Architectures and Network Programming Proceedings, 1999. OPENARCH'99. 1999 IEEE Second Conference on**. [S.l.], 1999. p. 90–97.

SINGH, R. et al. Run, walk, crawl: Towards dynamic link capacities. In: **Proceedings of the 16th ACM Workshop on Hot Topics in Networks**. New York, NY, USA: ACM, 2017. (HotNets-XVI), p. 143–149. ISBN 978-1-4503-5569-8. Available from Internet: <http://doi.acm.org/10.1145/3152434.3152451>.

SIVARAMAN, V. et al. Heavy-hitter detection entirely in the data plane. In: ACM. **Proceedings of the Symposium on SDN Research**. [S.l.], 2017. p. 164–176.

SMITH, J. M. et al. Switchware: accelerating network evolution (white paper). 1996.

SOMMERS, J.; BARFORD, P.; WILLINGER, W. A proposed framework for calibration of available bandwidth estimation tools. **Proceedings - International Symposium on Computers and Communications**, p. 709–718, 2006. ISSN 15301346.

SONG, H. Protocol-oblivious forwarding: Unleash the power of sdn through a future-proof forwarding plane. In: ACM. **Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking**. [S.l.], 2013. p. 127–132.

STRAUSS, J.; KATABI, D.; KAASHOEK, F. A measurement study of available bandwidth estimation tools. In: **Proceedings of the 3rd ACM SIGCOMM Conference on Internet Measurement**. New York, NY, USA: ACM, 2003. (IMC '03), p. 39–44. ISBN 1-58113-773-7. Available from Internet: <http://doi.acm.org/10.1145/948205.948211>.

TENNENHOUSE, D. L. et al. A survey of active network research. **IEEE communications Magazine**, IEEE, v. 35, n. 1, p. 80–86, 1997.

The P4 Language Consortium. P4 16 Language Specification v1.1.0. p. 129, 2018. ISSN 0279-1072. Available from Internet: <http://p4.org.>

WETHERALL, D. J.; GUTTAG, J. V.; TENNENHOUSE, D. L. Ants: A toolkit for building and dynamically deploying network protocols. In: IEEE. **Open Architectures and Network Programming, 1998 IEEE**. [S.l.], 1998. p. 117–129.

YANG, L. et al. **Forwarding and control element separation (ForCES) framework**. [S.l.], 2004.

ZHANG, E.; XU, L. Capacity and token rate estimation for networks with token bucket shapers. **Computer Networks**, Elsevier Ltd., v. 88, p. 1–11, 2015. ISSN 13891286. Available from Internet: <http://dx.doi.org/10.1016/j.comnet.2015.05.013>.

ZHU, Y. et al. Packet-level telemetry in large datacenter networks. In: ACM. **ACM SIGCOMM Computer Communication Review**. [S.l.], 2015. v. 45, n. 4, p. 479–491.