Partially Offloading Hierarchical Quality of Service to Commodity Hardware
Half Time Review Report

Rubens Figueiredo

December 10, 2024

1 Abstract

Broadband networks serve as the backbone of modern communication networks, linking users to network services. To expand markets and improve service quality, Internet Service Providers (ISPs) seek ways to make networks more flexible while reducing their capital (CAPEX) and operational (OPEX) costs. Network Function Virtualization (NFV) addresses this by deploying network functions as software on commodity hardware, typically using x86 CPUs and off-the-shelf Network Interface Cards (NICs). A critical Virtual Network Function (VNF) in broadband access, the Broadband Network Gateway (BNG), separates the provider's access network from the routed Layer 3 network, serving as a packet gateway. However, meeting requirements for user traffic, with tens of thousands of subscribers and multiple service classes per subscriber is challenging. High data rates and low latency are essential performance targets, but software-based network functions often fall short of the performance delivered by fixed hardware platforms. To improve performance, VNFs may be offloaded from the CPU to hardware, improving performance and preserving CPU resources. However, the hardware targets are typically less flexible or scalable.

This work proposes methods to improve BNG performance, particularly by proposing novel ways to split VNFs between software and hardware data paths. Indeed, *partial offloading*, combines the flexibility of software-based VNF with the performance guarantees of hardware platforms. Our primary VNF of focus is the QoS mechanism of the BNG, referred to as Hierarchical Quality-of-Service (HQoS), which enforces data rates and traffic isolation in a per-subscriber and service class basis. A key challenge in executing HQoS in software is the high CPU demand required to maintain fine-grained state, leading to increased costs for polling and tracking active queues.

Our contributions focus on the integration of commodity hardware for HQoS execution. The first main contribution of this work is a review and taxonomy of VNF benchmarking metrics and frameworks. The second contribution is a performance evaluation to identify performance bottlenecks in software HQoS. We performed experiments using various workloads, load distribution strategies, and platform configurations. Additionally, we analyze HQoS scalability by leveraging multiple CPU cores and NIC queues. Finally, the third main contribution focuses on full and partial offloading mechanisms. We design and evaluate different partial offloading strategies, examining their impact on Quality of Service (QoS) performance metrics and energy efficiency.

The outcomes of this work aim to advance the development of next-generation high-performance networks.

2 Introduction

As applications and network services become an essential to daily life, modern communication networks become critical infrastructure. Broadband networks play a central role in this infrastructure, especially as more users access the Internet through mobile devices. Moreover, recognizing broadband role in driving socioeconomic development [1], the European Commission has set ambitious goals to deliver gigabit internet access to every household until 2030 [2]. Thus, providing high-speed network access presents both a challenge and a business opportunity to Internet Service Providers (ISPs), who must keep capital (CAPEX) and operational (OPEX) costs low while expanding their markets and improving service quality [3]. Addressing this, Network Function Virtualization (NFV) has emerged as a promising cost reduction strategy, as it leverages commodity off-the-shelf hardware and software to deploy network functions. In particular, NFV transforms network functions into individual micro-services known as Virtual Network Functions (VNFs).

The Broadband Network Gateway (BNG) is a critical VNF in the broadband access network architecture, serving as the interface between the provider owned access network and the routed Layer 3 network. Broadband Forum (BBF) specifications TR-178 [4] and TR-459 [5] outline the requirements for BNGs, which include access line termination, bidirectional traffic forwarding, and Quality of Service (QoS) functions for traffic policing and rate-limiting [5]. Network service quality is measured by QoS metrics, including low latency and low latency deviation (jitter), with QoS functions prioritizing latency critical traffic like Voice-over-IP (VoIP) over lower priority, data-rate heavy traffic such as IPTV.

But as network services diversify, more distinct service classes emerge, each with their own QoS requirements. Beyond traffic class prioritization, subscriber traffic must also be distinguished to ensure accurate accounting and enforcing Service Level Assurances (SLAs). Moreover, subscribers may be grouped to meet specific QoS objectives. This kind of multi-level QoS enforcement is achieved through Hierarchical Quality-of-Service (HQoS) algorithms, which manage traffic by organizing schedulers into several levels and applying QoS policies to maintain rate limits and transmission priorities over all flows [6]. HQoS consists of a set of queues, scheduling algorithms and shapers, used together to implement fine-granular traffic control.

As network operators shift to NFV, the scale and complexity of deployments place significant demands on the VNFs. Meeting the performance targets of hundreds of gigabit per second, with low and guaranteed latency for thousands of users, presents specific problems including:

Problem 1: Performance limitations of commodity hardware. In HQoS, rate limiting a large amount of traffic classes presents a challenge due the inefficiency of maintaining one queue per rate limit, which increases CPU costs for polling and tracking active queues [7]. Scaling CPU-based VNFs involves using multiple cores and multiple Network Interface Card (NIC) queues. This software scaling process is not necessarily applicable when the algorithm requires shared access to the same data structures, as synchronization costs can become a bottleneck. Hence, running VNF as software applications introduces significant performance overheads.

Problem 2: Offloading full VNF to hardware. Hardware offloading of individual VNFs has emerged as a promising approach to improve performance. By executing the VNF on dedicated hardware platforms, hardware offloading accelerates computations and frees resources at the CPU [8]. Available hardware platforms include programmable or fixed function NICs, Field-Programmable Gate Array (FPGA) or Application Specific Integrated Units (ASIC), each offering unique benefits depending on the use-case. But fully deploying on commodity hardware is often unfeasible, as such hardware typically lacks resources, such as memory or the number of queues required for broadband network scale. For example, when considering 35000 subscribers, each with 8 traffic classes, such a scale require 280000 queues [3], far outpacing the number of queues available in commodity NICs. Hence, hardware offloads are constrained by the limited resources of hardware platforms. Moreover, hardware-based VNF implementations typically require substantial time for reprogramming, limiting the flexibility of the solution.

Problem 3: Splitting VNF over heterogeneous hardware offload targets. Selecting appropriate datapaths for HQoS execution is challenging due to the limitations of monolithic VNF implementations, whether in software or fully offloaded to hardware. These implementations often waste hardware resources, result in coarse-grained resource allocation, and hinder scalability [9]. To address these inefficiencies, VNF decomposition into smaller, reusable subagents has been proposed [10]. This approach identifies reusable packet processing primitives, such as match-action tables and parser trees [10]. While modern commodity NICs already support hardware primitives like queues, schedulers, and shapers, the decomposition of HQoS processes has yet to receive adequate attention.

Together, these limitations and design complexities highlight the challenges facing VNFs in achieving scalable, high-performance solutions on commodity hardware. This document outlines the progress and the future plans for a PhD thesis *Partially Offloading Hierarchical Quality of Service to Commodity Hardware*. This work tackles challenges of maintaining high-performance HQoS on commodity hardware by designing and evaluating *HQoS partial offloading*, a process that involves decomposing HQoS into the different primitives, and executing them across both software and hardware datapaths.

This document is organized as follows. Section 3 presents the background and relevant research works for this report. Section 4 identifies the research questions designed to lead the research work. Section 5 explains the research methodology used. Section 6 summarizes our answers to the research questions by presenting the research articles related to this work. Section 7 provides an overview of the various courses taken. the Section 8 gives a short description on the ethical nature of this work. Finally, Section 9 finalizes this report.

3 Literature Review

3.1 Background

This section provides background information on the broadband networks and QoS algorithms used throughout this report.

Internet access typically begins when a home gateway establishes a connection to the core network. The authentication and termination of the access line happens at the BNG, which then manage and forwards subscriber traffic [5]. Generally, the services of a BNG have been described in the BBF TR-178 [4]. Recently, accompanying the trend of Software Defined Networking (SDN), the control and data plane of the BNG have been disaggregated, as defined in the BBF TR-459 [5] and the RFC 8772 [11].

We now describe the various functions applied on the BNG data plane. Different operations are applied to the packets depending on the direction of the traffic, either *upstream* (from the subscribers to the core), or *downstream* (core to subscribers). To enable the operator to identify each subscriber's traffic, header encapsulations such as VLAN or Point-to-Point Protocol (PPP) are used, which need to be either added in the downstream or removed in the upstream. Other functions include Access Control Lists (ACL) to verify that incoming traffic is legitimate, accounting of subscriber usage of data plans, and IGMP/ MLD replication for IPTV Multicast.

The BNG also applies QoS functions to police and rate-limit traffic. While upstream traffic is generally simpler due to differences in volume and access patterns [3], downstream traffic is significantly larger, primarily driven by voice, video, and data services, collectively known as *triple play* [12]. Recently, service offerings have expanded to up to eight different services, to include video game streaming or private Virtual Private Networks (VPN), as defined in [13]. These services come with varying and sometimes conflicting QoS requirements, such as throughput, latency, or packet loss. Beyond individual services, different traffic demands between subscribers may increase latency when sharing the access network [3]. Hence, traffic from different subscribers and different services needs to be isolated by classifying it into different service classes and enqueueing it separately. This is achieved through a *classful* QoS algorithm, typically involving three main stages: 1. **Classify:** Incoming packets are classified into flows based on fields in the packet header. 2. **Enqueue:** Packets are placed in their designated destination queues. 3. **Dequeue:** A scheduling algorithm selects a queue, calculates transmission eligibility, and manages packet transmission. Figure 1 illustrates the steps of a generic classful QoS algorithm.

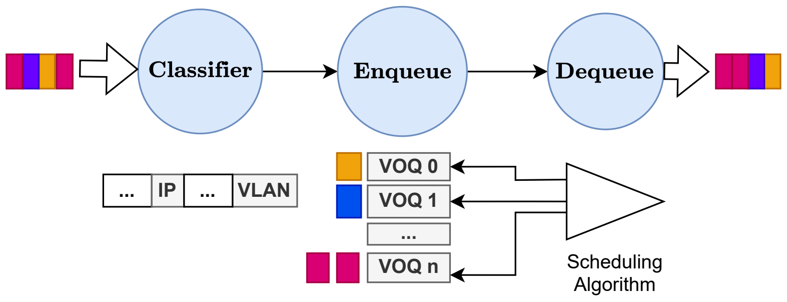
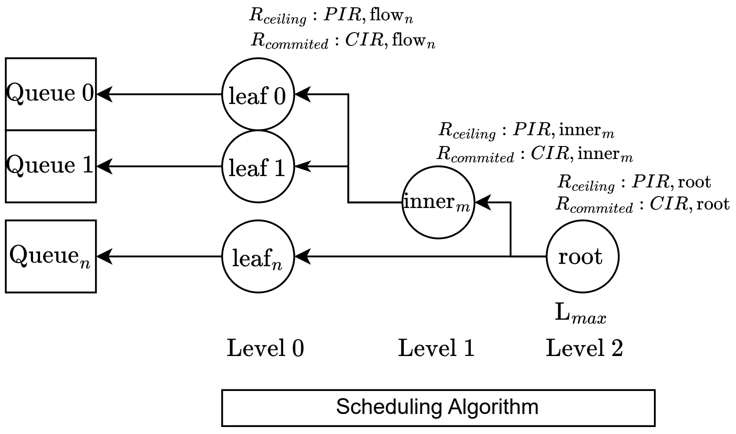


Figure 1: Illustration of a classful QoS algorithm.

To rate limit and apply transmission priorities for the flow queues, the scheduling algorithm visits each queue in an order defined by a scheduling algorithm, typically defined as strict priority (SP) mode or Weighted Round Robin (WRR). HQoS further organizes schedulers into levels, such as flow-group, subscriber, or network aggregation, according to the corresponding scheduling algorithm [6]. These levels consist of three types of nodes: leaf, inner, and root nodes. Leaf schedulers directly interface with the queues, and higher scheduler levels represent traffic aggregates, e.g. subscriber or data-center tenant traffic. Finally, the root scheduler typically represents the physical output port. Shapers can be implemented in any of these levels, where packets are selected for transmission based on the token-bucket status for the different traffic classes. In particular, srTCM [14] and drTCM [15] are used to enforce rate and burstiness limits. Figure 2 illustrates the HQoS scheduling tree.

Research on hierarchical schedulers has resulted in a wide array of different algorithms with several design goals and complexity tradeoffs. An ideal formulation based on a hypothetical fluid model of HQoS is realized by the Hierarchical Generalized Processor Sharing (H-GPS) [16]. Several real-world approximations of H-GPS have been proposed, including Hierarchical Packet Fair Queuing (H-PFQ) [16] or the Multiclass WRR [17]. In practice, these algorithms are typically evaluated by their implementation complexity, worst-case fairness, and delay bounds.

A considerable amount of literature has been published on the topic of network schedulers and their implementation efficiency. In the following section, we organize the literature into software and hardware contributions and identifies the primary contribution and the stage of the QoS targeted by the studied mechanism.



$R_{ceiling} : PIR, \text{flow}_n$
$R_{commited} : CIR, \text{flow}_n$
Queue 0
Queue 1
Queue$_n$
leaf 0
leaf 1
leaf$_n$
$R_{ceiling} : PIR, \text{inner}_m$
$R_{commited} : CIR, \text{inner}_m$
inner$_m$
$R_{ceiling} : PIR, \text{root}$
$R_{commited} : CIR, \text{root}$
root
L$_{max}$
Level 0
Level 1
Level 2
Scheduling Algorithm

Figure 2: Illustration of a two level HQoS scheduling tree.

3.2 HQoS Data-paths

3.2.1 Software

In this section, we describe some of the literature available on executing the HQoS as software entities.

A considerable amount of literature has been published in improving HQoS under Linux. Traffic control in Linux is available through 'tc'[1], offering several hierarchical schedulers, such as Enhanced Transmission Selection (ETS), which implements 802.1Qaz [18], the Hierarchical Token Bucket (HTB), or Class Based Queueing (CBQ). However, the performance evaluation of these hierarchical schedulers reveals challenges in achieving line rates of 100 Gbps [19]. Moreover, the Hierarchical Link Scheduler (HLS) [19] was proposed to address incorrect rate allocations in CBQ and HTB, due to not isolating rate guarantees to nodes in the inner branches of a class hierarchy. Implemented as a Linux qdisc, HLS was evaluated using a metric that measures deviation from the ideal rate allocation of an hierarchical bit-by-bit round-robin scheduler, and was compared to CBQ and HTB. Despite improvements, it showed similar performance bottlenecks as the other schedulers when evaluating with up to 1000 leaf classes.

Indeed, a common challenge in achieving fine-grained rate limiting is the inefficiency of maintaining one queue per rate limit [7]. Carousel [7] addresses this inefficiency by using Timing Wheels, which are time-indexed queues that maintain shaper pacing with only one queue per rate limit. The process scales by adding more independent Timing Wheels instances on parallel cores. To synchronize aggregate rates, the framework adjusts the timer wheels across CPUs to maintain work-conservation and max-min fairness. However, since Timing Wheels are timestamp-based, they do not support strict priorities.

In [20], one proposal to execute HQoS in a multi-core manner is described, aiming to improve the performance of the Linux Kernels' HTB. The approach separates the enqueue and dequeue operations in independent cores, bypassing read and write operators to shared tree structures. This effectively calculates scheduling eligibility determined during dequeue, thereby avoiding the need for locking mechanisms. The systems' throughput and number of queues was evaluated through experiments, assessing the capability to shape to two types of services. While our work differs in its focus on user space networking, the proposal of decoupling the enqueue and dequeue

[1]https://tldp.org/HOWTO/Traffic-Control-HOWTO/intro.html

mechanism might be applicable to the DPDK scheduler, potentially improving performance at the expense of CPU cores.

Eiffel [21] decouples the Timing Wheels and priority queues to build a software hierarchical scheduler. Packets are enqueued first to the Timing Wheel queue, and then to the hierarchical scheduler, effectively decoupling shaping and scheduling. Moreover, Eiffel proposes to use approximate priority queues, with the use of the hardware instruction Find First Set (FFS) for fast lookup.

Addressing inflexible NIC schedulers, Loom [22] proposes a software NIC to support hierarchical scheduling and shaping operations. The Loom design applies a scheduler and shaping from multi-threaded applications, and offloads the per-flow scheduling to the NIC. Loom enqueues metadata along with the packet descriptor on the NIC queues, reducing the memory fetch overhead and allowing the packet to be scheduled without accesses from main memory.

3.2.2 Hardware

In this section, we describe some of the literature available on executing the HQoS on the hardware platforms.

Modern NICs contain hardware functionalities, offering features such as Receive-Side Scaling (RSS), multiple Rx and Tx queues and transmit steering (XPS), that are designed to enhance scalability and performance [23]. By executing certain functions, such as segmentation offloads and classifiers, directly in hardware, load is alleviated on the CPU, freeing up resources for more computationally expensive tasks. Despite their relatively closed nature, given that commodity NICs are typically fixed-function devices and only expose configuration parameters, these platforms remain attractive due to their low cost and seamless integration with standard hardware, making them viable candidates for offloading tasks.

Building on these capabilities, research has explored leveraging multi Tx queues to approximate hierarchical scheduling, as demonstrated in TONIC [24]. TONIC aims to ensure flow priority and fairness between different tenants running on a Linux environment by employing two mechanisms. First, Weighted Round Robin (WRR) is implemented by assigning multiple Tx queues to each tenant, where the weights are represented by the number of queues. Second, by enqueueing packets to the front of the queue, it ensures strict priority between tenants. By only changing the enqueue mechanisms, approximate hierarchical scheduling can be implemented on multi queue nics. However, this approach does not address traffic shaping, leaving open the question of whether shaping could be supported in such a setup.

Meanwhile, Kuperman et al. [25] take hardware offloading further by a hardware-based implementation of Linux's HTB scheduler. This solution programs the parameters of the Mellanox NIC directly through Linux's 'tc', moving entire HQoS enforcement to hardware. However, the primary limitation of this approach is the fixed limit of hardware queues which limits the scalability of the solution.

Moving packet scheduling the smartNIC world, FlowValve+ [26] exposes the multiple Tx queues as single hardware FIFO. Flow traffic is associated with a counter that tracks the traffic rate per flow, dropping the packets when enqueueing to the Tx queues.

One notable development has been the introduction of programmable scheduling. This builds from the observations that scheduling algorithms decide on the order and timings of packet transmissions, and that this information can be determined during the enqueue step [27]. Programmable schedulers expresses the scheduling algorithm as a *scheduling transaction*, where users can define a strategy for computing the rank of a packet, which is used to the position of the packet inside a Push-In First-Out queue. Hierarchical schedulers can be built using scheduling trees, built made of multiple PIFO queues [27]. However, the PIFO model was limited by the lack of shared shapers or short-term rate-limit guarantees.

4 Research Questions

To understand the role that partial offloading plays in the deployment of VNF, we outline the research questions designed to guide our study.

RQ 1 How can the BNG control plane interface be redesigned to support multiple target dataplanes, while reducing integration efforts for new platforms and functions?

The BNG can be deployed in multiple dataplanes, either software or hardware. From a configuration perspective, supporting multiple datapaths requires the ability to translate generic configuration data into datapath-specific language. However, different datapaths often use distinct APIs to configure various functions, resulting in heterogeneous and inconsistent deployments. To enable consistent deployments across datapaths, network resources must be abstracted, with key information synthesized and serialized using standardized data models.

RQ 2 What frameworks are used for benchmarking VNF performance, and what metrics are employed to characterize their performance?

We investigate the methodologies applicable to studying VNF performance. In particular, we focus on test-based experimentation, as it involves empirically evaluating performance under precisely defined conditions and workloads on real systems rather than models [28]. This is a crucial yet complex task due to highly heterogeneous execution environments, where the infrastructure can include different system configurations, testing workloads, and resource sharing conditions [29].

RQ 3 What is the impact of lower-priority traffic on the latency of delay-sensitive, high-priority traffic in a software multi-core BNG?

To understand the impact of QoS enforcement on subscriber traffic, we have defined a thorough experimental protocol. Generally, in software-based HQoS, an increase in the number of queues to poll, can impact latency [7]. To further explore this effect, we measure the effects of different workload configurations on the latency, specifically more users or service classes, as the specifics of this latency increase are not well documented. We evaluate the impact of congestion in lower-priority classes on high-priority traffic.

RQ 4 How do different load distribution strategies affect the latency of high-priority traffic?

Next we analyze the scalability of the HQoS process. The scheduler we are studying statically allocates overall port data rates to individual threads, as the state of the HQoS process is not shared among threads, leading to process isolation. Implementing shared access to the queue and scheduler structures would require locking mechanisms or lock-free data structures, both of which negatively impact performance [30]. The static isolation makes the software scheduling architecture static and less responsive to dynamic traffic scenarios, which raises the need for a mechanism to distribute traffic according to their QoS requirements. Achieving proper scalability demands, among others, dynamic load balancing between instances and flow affinity to CPUs to minimize cache misses.

RQ 5 How can commodity hardware hierarchical schedulers serve as effective targets for HQoS offloading?

We integrate and evaluate hardware data paths into the software scheduler, focusing on the hierarchical schedulers available in commodity NICs. This approach requires the integration of both software and hardware data paths, so we evaluate how this hybrid setup influences software scheduler operation and overall network performance.

RQ 6 How does partial offloading improve the performance of the software scheduler when deploying complex VNFs, such as HQoS?

After studying the software and hardware data-paths, we proceed to look into the challenges of partially offloading HQoS. With this research question we investigate methods that enforce fine-grained user rate shaping over various isolated HQoS threads. We evaluate the benefits of partial offloading on the latency and throughput of the processed traffic, as well as the required energy consumption.

5 Research Methodology and Methods

5.1 Research Methods

In this section we provide an overview of the different research methods used during this work.

Survey of Available Literature In order to identify and map the available literature, we have conducted a survey of the available materials on VNF performance benchmarking. We have filtered the IEEE Xplore database for relevant papers. The survey is a scoping review [31] on the field of metrics, frameworks and workloads typically used on performance benchmarking experiments.

Performance Benchmarking Computer science has close ties with mathematics, providing the theoretical foundations, and engineering, providing the practical realization of computers. In empirical software engineering, measurement-based approaches, such as benchmarks, compare different hypotheses, methods, techniques and tools [28]. Benchmarking requires several steps, as the definition of measurement methodologies or workload selection. Given our focus on testing HQoS across both software and hardware, we aimed for a unified methodology applicable to both scenarios. A significant part of our work has been devoted to developing a comprehensive benchmarking methodology, derived from insights in previous literature.

Design Iteration and Testing Research is an iterative process that evolves over time, and this work reflects that journey. Leveraging the performance benchmarking methodology early in the process has been instrumental in identifying the performance characteristics and challenges of the initial HQoS design. Iterations on the design process is happening by developing and integrating new components, such as the partial offloading or the load distribution capabilities. Benchmarking these intermediate stages and evaluating their impact provide valuable insights to guide the development of this work.

6 Contributions

This section addresses the research questions and highlights the contributions of this thesis. Each contribution is organized according to the corresponding publication that explores and substantiates it. Figure 3 illustrates the VNF data-path, emphasizing each contribution and its alignment with the relevant research question.

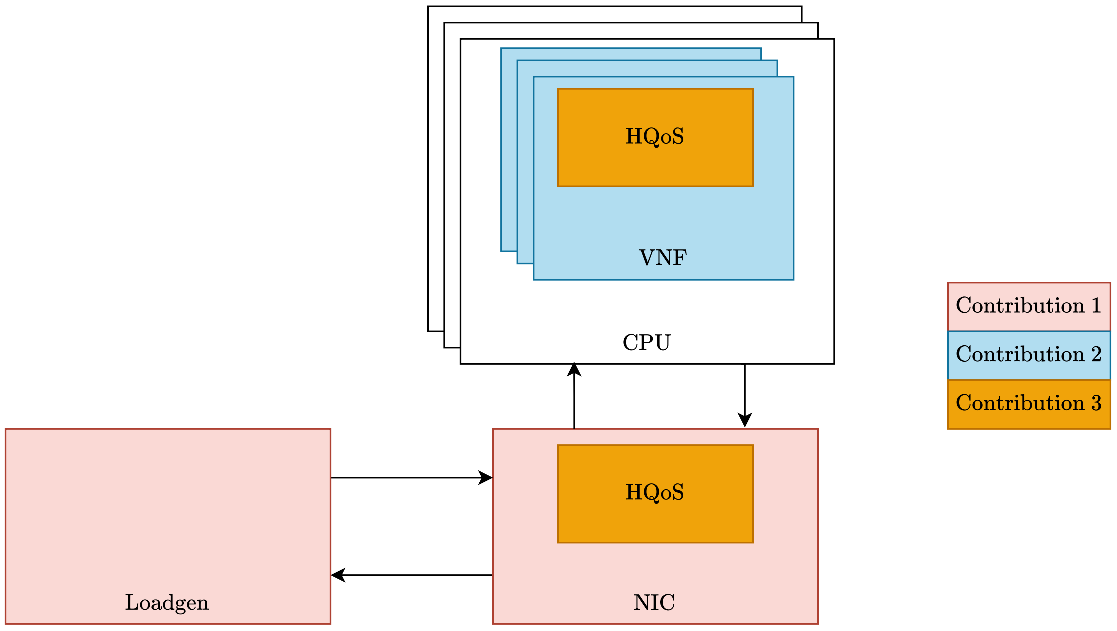


Figure 3: Illustration of the VNF data path in this work, with the primary contributions identified.

6.1 Contribution 1: VNF Performance Benchmarking

Paper 1: Performance Benchmarking of Virtual Network Functions – A survey

Status: Ongoing

Planned Submission: Q3 2025

Research Questions: RQ2

Summary and Connection To Research Questions: Benchmarking VNFs is complex due to the diverse system parameters, NFV execution environments, and workloads VNF. Therefore, it is essential to understand how different hardware/software combinations influence Key Performance Indicators (KPIs) such as throughput, latency, and energy efficiency. The research question is addressed by categorizing performance metrics and identifies framework characteristics that facilitate the orchestration of benchmarking experiments.

Lessons Learned: This paper surveys research on VNF benchmarking, focusing on key performance metrics and frameworks for their collection and analysis. We classify metrics by their insights into VNF-internal states and execution environments, discuss their application for evaluating VNF efficiency, and outline essential features of effective benchmarking frameworks.

6.2 Contribution 2: Commodity Hardware High-performance VNF

Paper 2: BNG-HAL: A Unified API for Disaggregated BNGs [32]

Status: Accepted @ 2021 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)

Research Questions: RQ1

Summary and Connection To Research Questions: To answer the research question, we have proposed a BNG Hardware Abstraction Layer (BNG-HAL), to simplify the complexity BNG management by presenting a set of features that removes the dependency on specific protocols and hardware APIs. This approach consolidates all necessary operations for management and state control, translating them to the relevant hardware targets.

Lessons Learned: A Hardware Abstraction Layer (HAL) implements an abstract forwarding protocol across diverse hardware targets, even when specific protocols are unsupported [33]. This research has guided our focus towards the BNG, due to the design of an interface that enables BNG configuration within a single control plane.

Paper 3: Quality of Service Performance of Multi-Core Broadband Network Gateways [34]

Status: Accepted @ 2024 8th Network Traffic Measurement and Analysis Conference (TMA)

Research Questions: RQ3

Summary and Connection To Research Questions: To answer the impact of background traffic on high-priority traffic, this study presents performance benchmark of VPP/DPDK implementation of the BNG. We observed how background traffic increases latency and jitter for delay-sensitive traffic. We demonstrated how batch configuration and multi-core scaling may be used to counteract the latency increase.

Lessons Learned: This paper highlights the suitability of software BNG for guaranteeing performance. These observations motivate a mechanism that distributes traffic according to their QoS requirements. A traffic distribution method must ensure balance load between HQoS, in order to balance latency and throughput. Moreover, the NIC hardware queues must also be managed, enforcing another level of scheduling on the NIC.

Paper 4: Virtualized Hierarchical Quality of Service Performance Measurements

Status: Ongoing

Research Questions: RQ3, RQ4

Planned Submission: Q2 2025

Summary and Connection To Research Questions: In this paper we extend the analysis from Paper 2 by investigating the impact of background traffic on the latency of high-priority traffic, with a focus on how the number of queues affects latency of high priority traffic. We expand the benchmarking methodology done in the previous paper to provide an extensive characterization of the performance of the DPDK HQoS implementation. We additionally study the effects of congestion of low priority traffic and different load scaling mechanisms on the latency high priority traffic.

Lessons Learned: Increasing the number of queues impacts the latency of high-priority traffic. When low priority queues become congested, the scheduler quickly becomes overloaded, impacting as well the high-priority traffic stream. Our analysis of load-scaling mechanisms reveals various trade-offs, suggesting that the current HQoS software scheduler may be an incomplete solution for implementing scheduling in VNF-based HQoS.

6.3 Contribution 3: HQoS Partial Offloading

Paper 5: Network Interface Card Offloads Of Hierarchical Quality of Service

Status: Ongoing

Planned Submission: Q1 2026

Research Questions: RQ4, RQ5

Summary and Connection To Research Questions: In this study we analyze the impact of the NIC scheduler on the performance of the multi-core software HQoS. To answer the question, we evaluate different workload distribution strategies under different scheduler configurations. In particular, we evaluate the default WRR behavior of the NIC scheduler, which may result in increased latency of high-priority traffic. We additionally offload the strict priority scheduling to the NIC. This is the first study that fully looks into integrating the capabilities of the hardware scheduler and analyses HQoS partial offloading.

Planned Contribution: We integrate the hardware offload capabilities of the Intel E810 NIC on the DPDK software scheduler. This requires integrating device configuration functions to the software data-path.

Paper 6: Hierarchical Shapers On Multi-Core Platforms

Status: Planned

Planned Submission: Q3 2026

Research Questions: RQ5, RQ6

Summary and Connection To Research Questions: In this study, we address the limitations of the software shaper to work in a multi-core CPU. In particular, when distributing traffic classes across isolated HQoS instances, the user level rates will not be synchronized, leading to errors at the shaper level.

Planned Contribution: We enable HQoS to be executed in a multi-core scheduler, while enforcing fine-grained rate limits. This requires a mechanism between threads, that coordinates the user rate limits between threads. We evaluate the impact of the coordination mechanism in terms of the ability of the scheduler to coordinate user rates. To deal with the adaptive workloads, we propose an extension to Receive Side Scaling (RSS), that distributes the workload to NIC queues according to traffic classes. We evaluate the latency and energy consumption of the entire solution.

7 Course Planning

Table 1: List of courses taken and the progress achieved.

Course Name	**Phase**	**Date**	**ECTS**
Vetenskapsteori för doktorander	Finished	2024-08-22	4.5
Att nyttiggöra forskning och vetenskap	Finished	2024-08-22	4.5
Introduktion till forskarstudier i datavetenskap	Finished	2024-01-22	1.5
Forskningsetik för doktorander, grundkurs	Finished	2023-03-07	3.0
Vetenskapligt skrivande i naturvetenskap och teknik	Finished	2022-12-20	5.0
Introduction to statistical methods in science and technology	Finished	2022-03-28	4.0
Dataplansprogrammering	Finished	2022-02-08	4.5
DISCO Reading Course	Finished	2022-12-31	4.5
SIGS-CyberSec: Phd Course on Future Network Security	In progress	2025 VT	3
Computer Science Colloquium	In progress	2026 VT	1.5
Literature Study Course on NFV performance benchmarking	In progress	2025 VT	6
Peer reviewing in Computer Science	In progress	2026 HT	2
Standardization and Sustainability	Planned	2025 VT	6
Molecular Communications and Nanonetworks	Planned	2025 VT	6
Communicating Science	Planned	2026 VT	4.5
Sum of Points			60.5

8 Ethical Review

This work does not deal with any personal data, animal or raise otherwise ethical concerns. That being said, according to BEREC Guidelines [35] the enforcement of QoS must be based on objective technical quality of service parameters and metrics, and not based on commercial considerations.

9 Conclusion

Supporting the sustainable coexistence of new network services requires efficient use of resources within the data center. The micro-services approach of NFV works towards this goal, by mapping resources to the VNF. VNF decomposition and hardware offloading further advance this idea, likewise improving performance and flexibility. This report has outlined the progress made towards proposing, developing and evaluating this idea. Future work will focus on implementing and expanding upon the concepts and approaches discussed here.

References

[1] M. De Clercq, M. D'Haese, and J. Buysse, "Economic growth and broadband access: The European urban-rural digital divide," *Telecommunications Policy*, vol. 47, p. 102579, July 2023.

[2] "COMMUNICATION FROM THE COMMISSION TO THE EUROPEAN PARLIAMENT, THE COUNCIL, THE EUROPEAN ECONOMIC AND SOCIAL COMMITTEE AND THE

COMMITTEE OF THE REGIONS 2030 Digital Compass: the European way for the Digital Decade," 2021.

[3] R. Kundel, L. Nobach, J. Blendin, W. Maas, A. Zimber, H.-J. Kolbe, G. Schyguda, V. Gurevich, R. Hark, B. Koldehofe, and others, "OpenBNG: Central office network functions on programmable data plane hardware," *International Journal of Network Management*, vol. 31, no. 1, p. e2134, 2021. Publisher: Wiley Online Library.

[4] "TR-178: Multi-service Broadband Network Architecture and Nodal Requirements," Technical Report TR-178, Broadband Forum, Sept. 2014. https://www.broadband-forum.org/technical/download/TR-178_Issue-1.pdf.

[5] "TR-459: Control and User Plane Separation for a disaggregated BNG," Technical Report TR-459, Broadband Forum, 2020. https://www.broadband-forum.org/technical/download/TR-459.pdf.

[6] F. Fejes, S. Nadas, G. Gombos, and S. Laki, "DeepQoS: Core-Stateless Hierarchical QoS in Programmable Switches," *IEEE Transactions on Network and Service Management*, vol. 19, pp. 1842–1861, June 2022.

[7] A. Saeed, N. Dukkipati, V. Valancius, V. The Lam, C. Contavalli, and A. Vahdat, "Carousel: Scalable Traffic Shaping at End Hosts," in *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, (Los Angeles CA USA), pp. 404–417, ACM, Aug. 2017.

[8] X. Fei, F. Liu, Q. Zhang, H. Jin, and H. Hu, "Paving the Way for NFV Acceleration: A Taxonomy, Survey and Future Directions," *ACM Computing Surveys*, vol. 53, pp. 1–42, July 2021.

[9] S. R. Chowdhury, Anthony, H. Bian, T. Bai, and R. Boutaba, "A Disaggregated Packet Processing Architecture for Network Function Virtualization," *IEEE Journal on Selected Areas in Communications*, vol. 38, pp. 1075–1088, June 2020. Conference Name: IEEE Journal on Selected Areas in Communications.

[10] H. Soni, M. Rifai, P. Kumar, R. Doenges, and N. Foster, "Composing Dataplane Programs with P4," in *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication*, (Virtual Event USA), pp. 329–343, ACM, July 2020.

[11] S. Hu, D. E. Eastlake 3rd, F. Qin, T. M. Chua, and D. Huang, "The China Mobile, Huawei, and ZTE Broadband Network Gateway (BNG) Simple Control and User Plane Separation Protocol (S-CUSP)," Request for Comments RFC 8772, Internet Engineering Task Force, May 2020. Num Pages: 124.

[12] N. Solihah and M. I. Nashiruddin, "Performance Evaluation of the 10 Gigabit Symmetric PON for Triple-Play Services," in *2020 IEEE International Conference on Communication, Networks and Satellite (Comnetsat)*, (Batam, Indonesia), pp. 136–143, IEEE, Dec. 2020.

[13] A. Nowicki, "AGF Functional Requirements," July 2023.

[14] J. Heinanen and R. Guerin, "A Single Rate Three Color Marker," Request for Comments RFC 2697, Internet Engineering Task Force, Sept. 1999. Num Pages: 6.

[15] J. Heinanen and R. Guerin, “A Two Rate Three Color Marker,” Request for Comments RFC 2698, Internet Engineering Task Force, Sept. 1999. Num Pages: 5.

[16] J. C. R. Bennett and H. Zhang, “Hierarchical Packet Fair Queueing Algorithms,” vol. 5, no. 5, 1997.

[17] H. Chaskar and U. Madhow, “Fair scheduling with tunable latency: A round-robin approach,” *IEEE/ACM Transactions on Networking*, vol. 11, pp. 592–601, Aug. 2003.

[18] S.-A. Reinemo, T. Skeie, and M. K. Wadekar, “Ethernet for High-Performance Data centers: On the New IEEE Datacenter Bridging Standards,” *IEEE Micro*, vol. 30, pp. 42–51, July 2010.

[19] N. Luangsomboon and J. Liebeherr, “HLS: A Packet Scheduler for Hierarchical Fairness,” in *2021 IEEE 29th International Conference on Network Protocols (ICNP)*, pp. 1–11, Nov. 2021. ISSN: 2643-3303.

[20] Z. Li, N. Yu, and Z. Hao, “A Novel Parallel Traffic Control Mechanism for Cloud Computing,” in *2010 IEEE Second International Conference on Cloud Computing Technology and Science*, (Indianapolis, IN, USA), pp. 376–382, IEEE, Nov. 2010.

[21] A. Saeed, Y. Zhao, N. Dukkipati, M. Ammar, E. Zegura, K. Harras, and A. Vahdat, “Eiffel: Efficient and Flexible Software Packet Scheduling,”

[22] B. Stephens, A. Akella, and M. M. Swift, “Loom: Flexible and Efficient NIC Packet Scheduling,”

[23] P. Shinde, A. Kaufmann, T. Roscoe, and S. Kaestle, “We need to talk about NICs,”

[24] G. Kim and W. Lee, “Network Policy Enforcement With Commodity Multiqueue NICs for Multitenant Data Centers,” *IEEE Internet of Things Journal*, vol. 9, pp. 6252–6263, Apr. 2022.

[25] Y. Kuperman, M. Mikityanskiy, and R. Efraim, “Hierarchical QoS Hardware Offload (HTB),” https://netdevconf.info/0x14/pub/papers/44/0x14-paper44-talk-paper.pdf.

[26] S. Xi, F. Li, L. Hu, X. Wang, and K. Ren, “FlowValve+: Multi-queue Packet Scheduling Framework on SoC-based SmartNICs,” *IEEE Transactions on Services Computing*, pp. 1–15, 2024. Conference Name: IEEE Transactions on Services Computing.

[27] A. Sivaraman, S. Subramanian, M. Alizadeh, S. Chole, S.-T. Chuang, A. Agrawal, H. Balakrishnan, T. Edsall, S. Katti, and N. McKeown, “Programmable Packet Scheduling at Line Rate,” in *Proceedings of the 2016 ACM SIGCOMM Conference*, (Florianopolis Brazil), pp. 44–57, ACM, Aug. 2016.

[28] W. Hasselbring, “Benchmarking as Empirical Standard in Software Engineering Research,” in *Evaluation and Assessment in Software Engineering*, (Trondheim Norway), pp. 365–372, ACM, June 2021.

[29] R. V. Rosa, C. Bertoldo, and C. E. Rothenberg, “Take Your VNF to the Gym: A Testing Framework for Automated NFV Performance Benchmarking,” *IEEE Communications Magazine*, vol. 55, pp. 110–117, Sept. 2017. Conference Name: IEEE Communications Magazine.

[30] "56. Quality of Service (QoS) Framework — Data Plane Development Kit 23.11.0 documentation." https://doc.dpdk.org/guides/prog_guide/qos_framework.html.

[31] Z. Munn, M. D. J. Peters, C. Stern, C. Tufanaru, A. McArthur, and E. Aromataris, "Systematic review or scoping review? Guidance for authors when choosing between a systematic or scoping review approach," *BMC Medical Research Methodology*, vol. 18, p. 143, Dec. 2018.

[32] R. Figueiredo and A. Kassler, "BNG-HAL: A Unified API for Disaggregated BNGs," in *2021 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, (Heraklion, Greece), pp. 116–119, IEEE, Nov. 2021.

[33] D. Parniewicz, R. Doriguzzi Corin, L. Ogrodowczyk, M. Rashidi Fard, J. Matias, M. Gerola, V. Fuentes, U. Toseef, A. Zaalouk, B. Belter, E. Jacob, and K. Pentikousis, "Design and implementation of an OpenFlow hardware abstraction layer," in *Proceedings of the 2014 ACM SIGCOMM workshop on Distributed cloud computing*, (Chicago Illinois USA), pp. 71–76, ACM, Aug. 2014.

[34] R. Figueiredo, H. Woesner, A. Kassler, and H. Karl, "Quality of Service Performance of Multi-Core Broadband Network Gateways," in *2024 8th Network Traffic Measurement and Analysis Conference (TMA)*, (Dresden, Germany), pp. 1–10, IEEE, May 2024.

[35] "BEREC Guidelines on the Implementation of the Open Internet Regulation," June 2020.