A Model-based Approach Towards Real-time Analytics in NFV Infrastructures

Raffaele Bolla, Roberto Bruschi, Franco Davoli and Jane Frances Pajo

***Abstract*—Network Functions Virtualization (NFV) has recently gained much popularity in the research scene for the flexibility and programmability that it will bring with the software implementation of network functions on commercial off-the-shelf (COTS) hardware. To substantiate its roll out, a number of issues (e.g., COTS' inherent performance and energy efficiency, virtualization overhead, etc.) must be addressed, in a scalable and sustainable manner. Numerous works in the scientific literature manifest the strong correlation of network key performance indicators (KPIs) with the *burstiness* of the traffic. This paper proposes a novel model-based analytics approach for profiling *virtualized* network function (VNF) workloads, towards real-time estimation of network KPIs (specifically, power and latency), based on an M^X/G/1/SET queueing model that captures both the workload burstiness and system setup times (caused by interrupt coalescing and power management actions). Experimental results show good estimation accuracies for both VNF workload profiling and network KPI estimation, with respect to the input traffic and actual measurements, respectively. This demonstrates that the proposed approach can be a powerful tool for scalable and sustainable network/service management and orchestration.**

***Index Terms*—NFV, M^X/G/1/SET queue, Real-time analytics, Workload profiling, Power estimation, Latency estimation**

I. Introduction

Few years back, network *softwarization* has been deemed fundamental towards sustaining the fifth generation mobile radio network (5G) vision. By integrating networking paradigms with state-of-the-art Information Technology (IT) services (i.e., virtualization on top of commercial off-the-shelf (COTS) hardware), it strives to overcome infrastructure *ossification*.

Network Functions Virtualization (NFV) – an emerging softwarization solution – explores the software implementation of network functionalities that would run on COTS hardware [1]. Such a paradigm grants customization and portability to *virtualized* network functions (VNFs) that would accelerate service innovation and facilitate seamless service support, while minimizing capital expenditures (CAPEX). Despite the numerous gains attainable with NFV, some operational issues that stem from the underlying COTS hardware and virtualization approach adopted need to be handled effectively and efficiently; otherwise, the operational expenditures (OPEX) that result in meeting future demands will prove to become unsustainable.

R. Bolla, R. Bruschi, F. Davoli and J. F. Pajo are with the Department of Electrical, Electronic and Telecommunications Engineering, and Naval Architecture (DITEN) of the University of Genoa, and with the National Laboratory of Smart and Secure Networks (S2N) of the Italian National Consortium for Telecommunications (CNIT) (e-mail: raffaele.bolla@unige.it, roberto.bruschi@unige.it, franco.davoli@unige.it, jane.pajo@tnt-lab.unige.it).

Contrary to the special-purpose hardware mostly deployed within classical network infrastructures, lower performance and energy efficiency are intrinsic to COTS hardware. While the Advanced Configuration and Power Interface (ACPI) specification [2] equips most – if not all – of it with power management mechanisms (e.g., Low Power Idle (LPI) and Adaptive Rate (AR)), power savings come in trade-off with performance degradation [3]. Moreover, virtualization typically adds extra layer(s) in the networking stack that result in additional processing delays, further lowering the performance. For a given amount of workload, VNFs may consume even more energy than their physical counterparts [4].

Furthermore, given the highly modular and customizable nature of the virtualized network architecture, coping with the ensuing management complexity entails automated configuration, provisioning and anomaly detection. The ETSI NFV Management and Orchestration (NFV-MANO) framework [5] designates these responsibilities to the virtual infrastructure manager (VIM) of the NFV infrastructure (NFVI). The VIM seeks to obtain performance and anomaly information about virtualized resources based on capacity/usage reports and event notifications, and then to manage them accordingly – yet usually, measurable data do not directly expose network key performance indicators (KPIs).

Starting from available and easily measurable performance monitor counters (PMCs) in Linux host servers, this paper tries to bridge this gap through a model-based analytics approach for real-time VNF workload profiling and network KPI (i.e., power and latency) estimation. Specifically, the contribution of this work is two-fold:

- a complete analytical characterization of the power- and performance-aware virtualized system, taking into account the inherent workload *burstiness*, and;
- a novel model-based analytics approach for profiling VNF workloads, towards the real-time estimation of the ensuing power consumption and system latency.

An initial version of this work has been presented in [6], in which various PMCs are evaluated for the black-box estimation of key statistical features of the VNF workload, considering a fairly general renewal model (M^X/G/1/SET queue [7]) that captures traffic burstiness and system setup times. In this extended version, we provide a complete analytical characterization of the M^X/G/1/SET queue, which includes power and latency models. This not only augments the capabilities of the VIM, but is also suitable for state-of-the-art dynamic resource and service provisioning approaches. Moreover, we present a new and more thorough experimental

analysis and validation of the models adopted.

The remainder of this paper is organized as follows. Firstly, some technological background and related work are presented in Section II. Section III then describes the system under test (SUT), giving details on key power- and performance-aware parameters foreseen to impact its behaviour. Section IV provides the analytical characterization of the different aspects of an M^X/G/1/SET queue; then, the key model parameters are exposed from available and easily measurable PMCs in Section V. Experimental results are presented in Section VI, and finally, conclusions are drawn in Section VII.

II. Background and Related Work

In this section, a brief background on the technological scenario is presented, along with some related work.

A. The ACPI Specification

Most – if not all – of the COTS hardware in today's market is already equipped with power management mechanisms through the ACPI specification. The ACPI exposes the LPI and AR functionalities at the software level through the *power* (C_x) and *performance* (P_y) states, respectively. The former comprise the active state C_0 and the sleeping states $\{C_1, \ldots, C_{\mathcal{X}}\}$, while the latter correspond to different processing performances $\{P_0, \ldots, P_{\mathcal{Y}}\}$ at C_0. Higher values of x and y indexes indicate deeper sleeping states and lower working frequencies and/or voltages, respectively. Sleeping states, although resulting in lower power consumptions, incur performance degradation due to the wakeup times, whereas reduced processing capacity increases the service times. It can be noted how LPI and AR have opposite effects on the burstiness of the traffic (i.e., the former clusters packets into bursts, while the latter smoothens the traffic profile). Joint adoption of both mechanisms does not guarantee greater savings [8] [9]; negative savings may even result with the naïve use of the ACPI [10]. The optimum configuration largely depends on the burstiness of the incoming traffic.

B. Performance vs Flexibility in NFV

Basically, the NFVI can employ various virtualization layer solutions towards the deployment of VNFs. This involves selection among (or mixing of) different virtualization techonologies, as well as their corresponding platforms and I/O technologies, which govern the overall performance and flexibility of the implementation [11]–[13].

In more detail, the performance yardstick in NFV is a set of network KPIs (e.g., power consumption, latency, response time, maximum throughput, isolation, mobility management complexity, instantiation time, etc.); this set varies (or at least the weight of each component does) with the application. Nonetheless, the overall performance is closely linked to the level of abstraction, and hence, to the virtualization overhead introduced in the chosen implementation. For instance, as described in [11], typical *hypervisor-based* solutions create isolated virtual machines (VMs) that are highly abstracted and flexible but with relatively high overhead, while *container-based* solutions create isolated guests (referred to as containers) that directly share the host operating system (OS), thus avoiding much of the overhead, but with a number of flexibility limitations (e.g., consolidation of heterogeneous VNFs, mobility support, etc.).

Other ways for reducing the virtualization overhead regard the handling of network I/O. A number of works in the scientific literature (e.g., [14]–[16], among others) consider technologies like Single Root I/O Virtualization (SR-IOV) and Intel's Data Plane Development Kit (DPDK), which bypass the OS network stack. However, this entails building a specialized network stack on applications that require one, and the device cannot be shared with other applications [13].

In this work, we focus on the power consumption and latency as network KPIs, and consider a traditional VM-based VNF implementation in order to minimize the dependence of the proposed approach on the virtualization and I/O technologies. Nevertheless, it can also be applied to container-based and bypass VNF implementations.

C. Modeling and Analytics of Network KPIs in NFV

A large part of the state-of-the-art software-level modeling approaches use machine learning (ML) techniques also based on measurable PMCs. For instance, numerous PMC-based power models have already been proposed at the VM and core levels [17] [18], while [19] explores correlations between application-level quality of service (QoS) parameters like throughput and response time with the power readings from Intel's Running Average Power Limit (RAPL) interface [20]. High levels of accuracies can be obtained with ML-based approaches, provided that the appropriate set of PMCs is considered and an extensive dataset is available for training.

Another well-known approach for modeling and analyzing telecommunications systems is the application of queueing theory principles and, more recently, it is being adopted in the context of NFV as well. Most of the works in the literature regard estimating the system or queueing latencies towards efficient (QoS-aware) network service provisioning, considering networks of queues to model interactions among service chain/virtual system/VNF components, with each component modeled as a (unique) queueing system [21]–[24]. Delving deeper into the infrastructure level, [25] takes into account the impact of interrupt coalescing (IC) in the Network Interface Card (NIC) on VNF performance (in terms of latency and packet loss), while [26] considers the burstiness of the VNF workloads (both incoming and aggregated) in the power modeling.

In this work, we adopt the queueing model considered in [26], analytically characterizing the different aspects of the system; starting from there, a model-based analytics approach that uses – and adds value to – available PMCs is proposed towards real-time VNF workload profiling, as well as power and latency estimation.

III. System Description

Considering that the system behaviour highly depends on the ACPI configuration of the host, as well as the virtualization and I/O technologies used in the VNF implementation, more details on these aspects are provided in the following sub-sections.

A. ACPI Configuration

For a given (C_x, P_y) pair, a number of power- and performance-aware parameters can be defined.

1) Power requirements: The instantaneous power requirements vary with the core's state. Specific values of idle (Φ_i) and active (Φ_a) power consumptions are associated with each available power and performance state, respectively.

Moreover, transitions between C_0 and C_x are not instantaneous; hence, the power consumed in these periods must also be taken into account. Since the average power consumption during sleeping transitions ($C_0 \rightarrow C_x$) approximates Φ_i, we only consider the wakeup transitions ($C_x \rightarrow C_0$) in this work. Particularly, the power spike in the latter is associated with a wakeup power consumption Φ_w that is approximately $2.5\Phi_a$, as pointed out in [27].

2) System latencies: The total delay experienced by packets can be broken down into contributions of different system operations. As packets arrive at the RX queue, NICs may wait either for some time interval (i.e., time-based IC) or some number of arrival events (i.e., frame-based IC) before raising interrupts to notify the core of pending work. Generally, such service requests can occur while the core is in idle or active mode; in the former, there is an additional setup period due to the wakeup and reconfiguration operations before the actual packet processing begins.

At the NIC level, we consider the time-based IC, for which we define the period τ_{ic}. At the core level, we consider two setup contributions (i.e., due to wakeup and due to reconfiguration), for which we define the periods τ_p and τ_r, respectively. For the Sandy Bridge EP platform, core wakeup latencies are in the order of nano/microseconds [28], yet power spikes during the wakeup transitions can last a bit longer [27]; in any case, the value of τ_p depends on the core's power state C_x. Once in active mode, the core then performs some reconfiguration operations; the value of τ_r depends on the core's performance state P_y. In the context of power and latency modeling, τ_p and $\tau_l = \tau_{ic} + \tau_p + \tau_r$ will be considered, respectively.

After the completion of the setup period, backlogged packets are suppose to be served exhaustively (considering that packets have been already transferred in the main memory via the standard Direct Memory Access (DMA)) with an average processing capacity μ, which corresponds to the operating energy point of the performance state P_y.

B. VNF Implementation

With the current ubiquity of Linux servers and x86 hardware with virtualization extensions, Kernel-based Virtual Machine (KVM) [29] – being the default virtualization infrastracture of Linux that is already integrated in the kernel – offers simplicity in the VNF deployment and mobility management. In this respect, we consider a KVM-based VNF running on a Linux host in this work, but the approach is also applicable to (or easily adaptable for) other platforms.

Particularly, as a full virtualization solution, KVM is able to run VMs with unmodified guest OSs. Guest networking is implemented by the user space process Quick Emulator (QEMU), as detailed in [30]. VM processes are allocated a certain number of vCPUs, each one seen as a physical CPU by the guest OS. Then, VNFs run as guest user space processes in the corresponding VMs. Fig. 1 illustrates the traditional KVM architecture for VNF implementation.

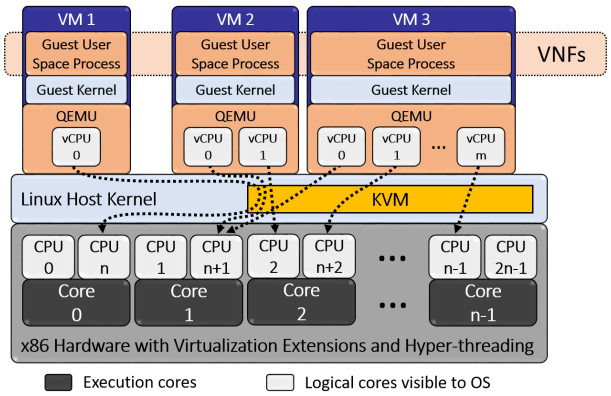


Fig. 1: Traditional KVM-based VNFs.

For simplicity, but without loss of generality, we suppose a one-to-one correspondence between the VM and the core to match the core workload, utilization, power consumption and latency with those of the VNF. More complex VNFs may consist of multiple VMs – each one running a VNF component (VNFC), but the overall performance can be derived from the individual performances of the components.

We note that with such a traditional VM-based implementation, switching between VNF and interrupt handler codes can be rather costly. To reduce this overhead, the interrupt and VM process affinities are set to different cores in this work. Like a pipeline model of some sorts, the core tasked with interrupt handling then notifies the one running the VM process via an inter-processor interrupt (IPI) for backlogged packets. Moreover, as also illustrated in [31], setting affinities or core pinning in such fashion improves the energy efficiency of the system.

Using the `ethtool` command [32], a number of parameters can be tuned in the NIC. In order to preserve as much as possible the shapes of the incoming/outgoing traffic of the VNF, we look into the IC and RX/TX ring parameter settings. For the former, we decided to keep the default settings since they are more or less equivalent with respect to the generated input traffic – specifically, options for adaptive IC (i.e., `adaptive-rx` and `adaptive-tx`) are off, the parameters for frame-based IC (i.e., `rx-frames` and `tx-frames`) are set to 0, and the parameters for time-based IC (i.e., `rx-usecs` and `tx-usecs`) are set to 3 μs and 0, respectively. On the other hand, the RX/TX ring buffer sizes are again set to the pre-set maximums (i.e., 4096) in order to maximize the NIC's ability to handle burst arrivals.

IV. Analytical Model

The energy-aware core hosting the VNF (or VNFC) is modeled as an M^X/G/1/SET queue, as in [6] [8] [9]. This model generalizes the well-known M^X/G/1 queue [33] for burst arrivals, by also covering the cases in which an additional setup period SET is necessary before service can be resumed.

In more detail, batches of packets arrive at the system at exponentially distributed inter-arrival times with a random batch size X. If the system is empty at the arrival instant,

TABLE I: Model notation.

Symbol	Description
λ	batch arrival rate
β_j	probability that an incoming batch is composed of j packets
$X(z)$	Probability Generating Function (PGF) of the batch size, $X(z) = \sum_{j=1}^{\infty} \beta_j z^j$
$\beta_{(k)}$	k-th factorial moment of the batch size, $\beta_{(k)} = E\{\frac{X!}{(X-k)!}\} = \lim_{z\to 1} \frac{d^{(k)}X(z)}{dz^{(k)}}$
$\tau(t)$	probability density of the setup time
$\tau^*(\theta)$	Laplace transform of $\tau(t)$
$\tau_{(i)}$	i-th moment of the setup time, $\tau_{(i)} = (-1)^i \frac{d^{(i)}\tau^*(\theta)}{d\theta^{(i)}}\big\vert_{\theta=0}$
$s(t)$	probability density of the packet service time
$s^*(\theta)$	Laplace transform of $s(t)$
$s_{(i)}$	i-th moment of the packet service time, $s_{(i)} = (-1)^i \frac{d^{(i)}s^*(\theta)}{d\theta^{(i)}}\big\vert_{\theta=0}$
μ	average packet service rate, $\mu = 1/s_{(1)}$
ρ	server utilization, $\rho = \frac{\lambda\beta_{(1)}}{\mu}$
$B^*(\theta)$	Laplace transform of the busy period density
$B_{(i)}$	i-th moment of the busy period, $B_{(i)} = (-1)^i \frac{d^{(i)}B^*(\theta)}{d\theta^{(i)}}\big\vert_{\theta=0}$
$P(z)$	PGF of the number of packets in the system at a random epoch
W	average waiting time in the queue

SET is initiated; service only begins after the completion of SET. Packets are queued as they arrive and served individually with generally-distributed service times S. Moreover, we approximate the loss probability in a queue with finite buffer N by the stationary probability that the number n of customers in the infinite-buffer queue at a generic time t be greater than N ($Pr\{n > N\}$). Therefore, hereinafter, we will consider the infinite buffer case.

More details on the different model components are presented in this section – from the arrival, setup and service processes, to key networking KPIs. The model notation is given in Table I.

A. Traffic Model

In telecommunications networks, where burst packet arrivals are more representative of the traffic behaviour rather than single arrivals, effectively capturing the burstiness is essential. The Batch Markov Arrival Process (BMAP) has long been established in this respect [34] [35]; BMAP allows for dependent and non-exponentially distributed packet inter-arrival times, while keeping the tractability of the Poisson process [36]. Starting from this, we suppose that packets arrive according to a BMAP with batch arrival rate λ.

To characterize the random batch size X, let β_j be the probability that an incoming batch is composed of j packets ($j = 1, 2, \ldots$). Then, the Probability Generating Function (PGF) of X is given by

$$X(z) = \sum_{j=1}^{\infty} \beta_j z^j \tag{1}$$

from which we obtain the first ($\beta_{(1)} = \sum_{j=1}^{\infty} j\beta_j$) and second ($\beta_{(2)} = \sum_{j=1}^{\infty} j^2\beta_j - j\beta_j$) factorial moments of the batch size. The offered load in packets per second (pps) is then obtained as $OL = \lambda\beta_{(1)}$.

Given λ, $\beta_{(1)}$ and $\beta_{(2)}$, the burstiness of the traffic can already be well estimated. However, a common difficulty stems from the fact that the discrete probability distribution $\{\beta_j, j = 1, 2, \ldots\}$ may not be given, and typically requires detailed analysis of packet-level traces [8]. As an alternative approach, we propose to estimate the factorial moments from easily measurable parameters (e.g., VNF workload, idle and busy times), which will be discussed in Section V.

B. Setup Model

With the deterministic nature of the setup period due to core wakeup transitions and reconfigurations considered in this work, the Laplace transform of the probability density $\tau(t)$ can be reduced to

$$\tau^*(\theta) = e^{-\tau\theta} \tag{2}$$

From this, the first and second moments of the setup time are simply given by $\tau_{(1)} = \tau$ and $\tau_{(2)} = \tau^2$, respectively, with

$$\tau = \begin{cases} \tau_p & \text{in the context of power consumption, and} \\ \tau_l & \text{in the context of latency.} \end{cases} \tag{3}$$

C. Service Model

Once setup is completed, the core starts to serve backlogged packets and remains busy until the system becomes empty. We suppose that the VNF (or VNFC) running on the core has deterministic service times.

1) Service process: Generally, multiple VMs may be consolidated on the same core, and the service process can be captured by a discrete set of service rates μ_m with corresponding probabilities π_m, $m \in \{1, \ldots, M\}$, where M is the number of VMs sharing the core. The Laplace transform of the probability density $s(t)$ is then obtained as

$$s^*(\theta) = \sum_m \pi_m e^{-\frac{\theta}{\mu_m}} \tag{4}$$

However, in the special case of one-to-one correspondence between cores and VMs, Eq. (4) reduces to $s^*(\theta) = e^{-\frac{\theta}{\mu}}$, giving the first and second moments of the service time as $s_{(1)} = 1/\mu$ and $s_{(2)} = 1/\mu^2$, respectively. Note that the core utilization due to actual packet processing is obtained as $\rho = OL/\mu$ (< 1 for system stability).

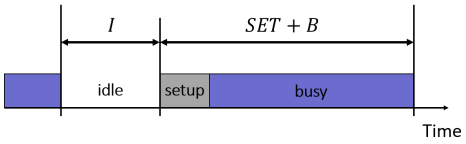


Fig. 2: A generic renewal cycle in a M^X/G/1/SET system.

2) Busy period distribution: From the derivations presented in Appendix A, the Laplace transform of the busy period density $B(t)$, specialized for deterministic setup and service times, is given by

$$B^*(\theta) = e^{-\lambda\tau(1-B^*_X(\theta))} X\left(e^{-\frac{\theta+\lambda-\lambda B^*_X(\theta)}{\mu}}\right) \tag{5}$$

from which we obtain the first ($B_{(1)} = \frac{(\beta_{(1)}/\mu)+\rho\tau}{1-\rho}$) and second ($B_{(2)} = \frac{(\beta_{(1)}+\beta_{(2)})(1+\lambda\tau)}{\mu^2(1-\rho)^3} + \frac{\rho\beta_{(1)}(2\tau+\lambda\tau^2)}{\mu(1-\rho)^2}$) moments of the busy periods. It is important to note that these expressions are particularly useful for estimating the power consumption and system latency, as we shall see further on.

D. Power Model

We adopt the power consumption model proposed in [26] for an energy-aware core running VMs. The model works according to a renewal process, where the idle (I) and delay busy ($SET + B$) periods constitute independent and identically distributed (iid) "cycles" (R), as illustrated in Fig. 2. A delay busy period, as defined in [7], starts with the arrival of the batch initiating the setup, and ends with the depature of the last packet in the system.

Based on classical renewal theory principles, the steady-state behavior of the stochastic process can be studied by looking at a representative cycle [33]. With this in mind, the average power consumption of the core is expressed as a sum of the average contributions incurred during the idle, setup (due to wakeups) and busy periods, $\Phi = \frac{I_{(1)}}{R_{(1)}}\Phi_i + \frac{\tau_{(1)}}{R_{(1)}}\Phi_w + \frac{B_{(1)}}{R_{(1)}}\Phi_a$, where $R_{(1)} = I_{(1)} + \tau_{(1)} + B_{(1)}$ is the average length of a renewal cycle, and $I_{(1)}$ is the average length of an idle period. Specializing these to the case of BMAP arrivals (i.e., $I_{(1)} = \frac{1}{\lambda}$), and deterministic setup and service times (i.e., $\tau_{(1)} = \tau$ and $B_{(1)} = \frac{(\beta_{(1)}/\mu)+\rho\tau}{1-\rho}$), we obtain $R_{(1)} = \frac{1+\lambda\tau}{\lambda(1-\rho)}$; then, with $\tau = \tau_p$

$$\Phi = \frac{1-\rho}{1+\lambda\tau_p}\Phi_i + \frac{\lambda\tau_p(1-\rho)}{1+\lambda\tau_p}\Phi_w + \rho\Phi_a \tag{6}$$

E. Latency Model

The system latency D is derived as the sum of the average waiting time W of a packet in the queue and its average service time (i.e., $s_{(1)} = 1/\mu$).

In more detail, Little's law defines the former as: $W = L/\lambda\beta_{(1)}$, where L is the average length of the queue that can be derived from the PGF $P(z)$ of the number of packets in the M^X/G/1/SET system at a random epoch, as $L = \frac{dP(z)}{dz}\big|_{z=1} - \rho$. By specializing the general expression for $P(z)$ presented in Appendix B to the case of deterministic setup and service times, we obtain

$$P(z) = \frac{1-X(z)e^{-\lambda\tau(1-X(z))}}{(1+\lambda\tau)\big(1-X(z)\big)} \frac{(1-\rho)(1-z)e^{-\frac{\lambda(1-X(z))}{\mu}}}{e^{-\frac{\lambda(1-X(z))}{\mu}} - z} \tag{7}$$

whence,

$$W = \frac{\rho\beta_{(1)} + \beta_{(2)}}{2\beta_{(1)}\mu(1-\rho)} + \frac{2\tau + \lambda\tau^2}{2\beta_{(1)}(1+\lambda\tau)} \tag{8}$$

and with $\tau = \tau_l$

$$D = \frac{\rho\beta_{(1)} + \beta_{(2)}}{2\beta_{(1)}\mu(1-\rho)} + \frac{2\tau_l + \lambda\tau_l^2}{2\beta_{(1)}(1+\lambda\tau_l)} + \frac{1}{\mu} \tag{9}$$

It is interesting to note that Eq. (8) can also be derived starting from the Laplace transform of the waiting time density, as in [37].

V. Exposing Model Parameters

With the M^X/G/1/SET queue as a basis, here we expose the key model parameters starting from available and easily measurable PMCs in the Linux host, in effect profiling the VNF workloads. Starting from this, the corresponding power consumption and system latency can then be readily derived from Eqs. (6) and (9), respectively.

A. Performance Monitor Counters

Linux has different utilities for performance monitoring – among them, the PMCs described in the following are considered in this work; other PMCs used in [6] did not work well with the BMAP emulation. Note that in the syntax of these utilities, the term 'CPU' refers to a core (or logical core, in the case of hyperthreading).

1) Idlestat: As a tool for CPU power/performance state analysis, the `idlestat` command [38] in trace mode is able to monitor and capture the $C-$ and $P-$ state transitions of CPUs over a user-defined interval. To run it in trace mode, the `--trace` option is used together with the `<filename>` and `<time>` parameters to specify the trace output filename and the capture interval in seconds, respectively. With the `-c` and `-p` options, $C-$ (including the $POLL$ state, in which the CPU is idle but did not yet enter a power state) and $P-$ states statistics are reported in terms of the time spent in each state per CPU.

2) VnStat: As a network traffic monitoring tool, the `vnstat` command [39] is able to report how much traffic (in terms of average rates) goes through a specific interface over a user-defined interval. This is done by using the `-tr` option together with the `<time>` parameter to specify the monitoring interval in seconds, and the `-i` option together with the `<interface>` parameter to specify the interface.

B. Estimation with PMCs

In this sub-section, we seek to expose the model parameters from the considered PMCs, for a given (C_x, P_y) pair.

1) Offered load and utilization: Measuring OL with the `vnstat` command is straightforward, as it corresponds to the rate of incoming traffic on the network interface bound to the VM process.

On the other hand, the utilization measurable with the `idlestat` command as

$$\tilde{\rho} = \frac{\tilde{T}_{P_y}}{\tilde{T}_{P_y} + \tilde{T}_{C_x} + \tilde{T}_{POLL}} \tag{10}$$

where $\tilde{T}_{P_y}$, $\tilde{T}_{C_x}$ and $\tilde{T}_{POLL}$ are the measured average times spent in the corresponding states, encompasses all operations (including reconfigurations, context switching, sleep transition, etc.) performed by the active core. While this gives indications on the utilization overheads incurred in this VNF implementation, we prefer to estimate the utilization due to actual packet processing, for which we consider

$$\hat{\rho} = \tilde{OL}/\mu \tag{11}$$

where $\tilde{OL}$ is the offered load measured with the `vnstat` command.

2) Batch arrival rate: By considering exponentially distributed inter-arrival times, λ can be estimated from the average idle times measurable with the `idlestat` command. Theoretically, $\tilde{I}_{(1)} \approx \tilde{T}_{C_x} + \tilde{T}_{POLL}$; however, with the high variance observed on $\tilde{T}_{POLL}$, we propose to consider only $\tilde{T}_{C_x}$ for stable estimates. Linear regression is then used to compensate for the discrepancy, giving

$$\hat{\lambda} = \frac{\alpha_1}{\tilde{T}_{C_x}} + \alpha_0 \tag{12}$$

where α_1 and α_0 are the computed regression coefficients.

3) Factorial moments of the batch size: Given $\tilde{OL}$ and $\hat{\lambda}$, $\beta_{(1)}$ (or the average batch size) can be directly estimated by definition as

$$\hat{\beta}_{(1)} = \frac{\tilde{OL}}{\hat{\lambda}} \tag{13}$$

While estimating $\beta_{(2)}$ is a bit more involved, it is essential for estimating D. In this regard, we propose to start from the expression of the second moment of the busy times $B_{(2)}$ (see Sub-section IV-C2), and consider

$$\hat{\beta}_{(2)} = max\left(0, \left(\frac{\mu^2}{1+\hat{\lambda}\tau_l}\right)\left(\hat{B}_{(2)}(1-\hat{\rho})^3 - \frac{\hat{\beta}_{(1)}(1+\hat{\lambda}\tau_l)}{\mu^2} - \frac{\hat{\rho}(1-\hat{\rho})\hat{\beta}_{(1)}(2\tau_l + \hat{\lambda}\tau_l^2)}{\mu}\right)\right) \tag{14}$$

with the max function ensuring $\hat{\beta}_{(2)} \geq 0$. Now we are left with how to obtain $\hat{B}_{(2)}$, for which we adopt the well-known theorems in statistics regarding the mean and variance of sample means.

In more detail, let $\{B^1_{(1)}, B^2_{(1)}, \ldots, B^\eta_{(1)}\}$ be a random sample of size η obtained from a busy period distribution with mean $B_{(1)} = E\{B\}$ and variance $var(B) = B_{(2)} - (B_{(1)})^2$. Supposing that $B^n_{(1)}$, $n = 1, \ldots, \eta$, are iid, we consider the standard estimators $\hat{B}_{(1)} = \frac{1}{\eta}\sum_{n=1}^{\eta} B^n_{(1)}$ and $\hat{var}(B_{(1)}) = \frac{1}{\eta-1}\sum_{n=1}^{\eta}\left(B^n_{(1)} - \hat{B}_{(1)}\right)^2$, and estimate the variance of the busy times as $\hat{var}(B) = \Delta t \hat{var}(B_{(1)})$ starting from sample means [40], where Δt is the observation period over which each sample of $B^n_{(1)}$ is obtained. Then, we can estimate the second moment of the busy times as

$$\hat{B}_{(2)} = \Delta t \hat{var}(B_{(1)}) + (\hat{B}_{(1)})^2 \tag{15}$$

In this work, we consider $\tilde{B}_{(1)} \approx \tilde{T}_{P_y} - \Delta T$, where ΔT is the busy overhead due to operations other than actual packet processing (which includes τ_r, context switching, sleep transitions, etc.). Eq. 15 is then applied on the set of samples $\{\tilde{B}^n_{(1)}, n = 1, \ldots, \eta\}$ to obtain an estimate of $B_{(2)}$.

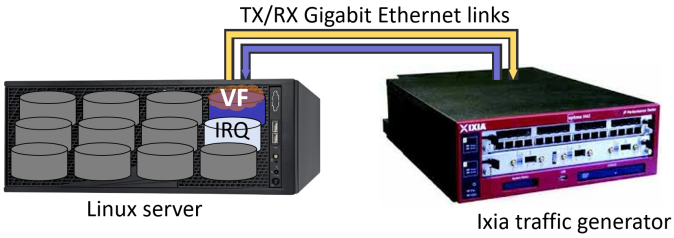


Fig. 3: Experimental testbed.

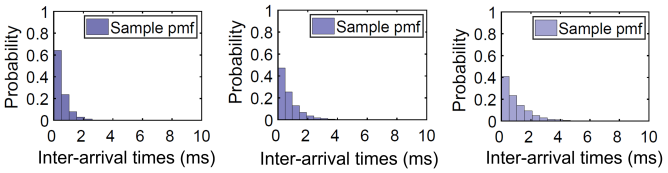
VI. Experimental Results

The proposed approach is evaluated considering a SUT equipped with two Intel® Xeon® E5-2643 v3 3.40GHz processor packages, running an OpenWrt [41] virtual firewall (VF). The latter is pinned (or with affinity set) to a single core and the interrupt request (IRQ) handling to another one. The SUT is connected via RX/TX Gigabit Ethernet links to an Ixia NX2 traffic generator, as shown in Fig. 3. The setup creates a controlled environment that allows monitoring of the SUT's PMCs, power consumption and system latency, as the burstiness of incoming traffic is varied.

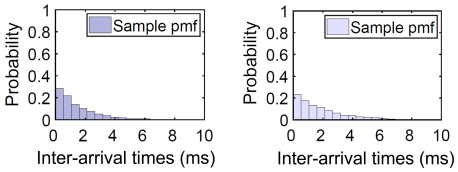
The ACPI configuration of the pinned core is set to (C_{1E}, P_{0T}) to maximize the system throughput, where the power state C_{1E} corresponds to the *Enhanced Halt* – the lightest sleeping state with improved power requirements, and the performance state P_{0T} to the maximum turbo frequency; while the rest of the cores are put to deep power saving state C_6. Under this configuration, we approximate the values of the following model parameters as: $\mu \approx 199628$ pps, $\tau_{ic} \approx 3$ μs, $\tau_p \approx 10$ μs, $\tau_r \approx 10$ μs, $\Phi_i \approx 8.33$ W, $\Phi_a \approx 53.38$ W and $\Phi_w \approx 133.45$ W.

In more detail, the processing capacity is estimated as the maximum system throughput measured from the user interface of the traffic generator. Setup components are derived from the IC configuration, wake-up latencies specified in the kernel's `cpuidle sysfs` and the results of [42]. Power related parameters are estimated starting from actual package-level (i.e., core part) measurements obtained with the `turbostat` command [43] – one of the many tools that expose power measurements from Intel's RAPL interfaces; [44] confirms from extensive tests that RAPL exposes true averages that are updated at fine-grained intervals.

Input traffic (composed of 64-byte Ethernet frames) is generated from the Ixia traffic generator by considering $1/\lambda \in \{0.5, 0.75, 1, 1.5, 2\}$ ms and $\beta_{(1)} \in \{1, 6, 12\}$ packets; details on the BMAP emulation approach are described in the following sub-section.

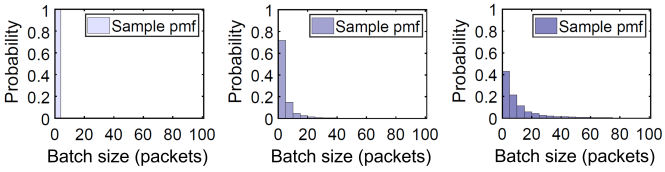


(a) $1/\lambda \approx 0.5$ ms (b) $1/\lambda \approx 0.75$ ms (c) $1/\lambda \approx 1$ ms



(d) $1/\lambda \approx 1.5$ ms (e) $1/\lambda \approx 2$ ms

Fig. 4: Distributions of the input batch inter-arrival times.



(a) $\beta_{(1)} \approx 1$ packet (b) $\beta_{(1)} \approx 6$ packets (c) $\beta_{(1)} \approx 12$ packets

Fig. 5: Distributions of the input batch sizes.

Lastly, the results obtained with the model-based approach are validated with respect to the inputs (for the workload profiling) and actual measurements (for the network KPI estimation); for each test point, 100 samples are collected, from which the 99% confidence intervals are obtained and indicated with error bars.

A. Emulating BMAP Arrivals

Rather than simply tuning deterministic parameters in the Ixia NX2 traffic generator, as in [6], incoming traffic is more accurately generated to emulate BMAP arrivals in this extended version. `Tcl` scripts are used to specify batch inter-arrival times and batch sizes. Realizations of the batch inter-arrival times are drawn from exponential distributions (setting the desired mean value), while those of batch sizes from truncated generalized Pareto distributions (with default *shape* parameter, and varying the *scale* and *location* parameters to approximate the desired mean value). The resulting pdfs of the inter-arrival times and batch sizes are illustrated in Figs. 4 and 5, respectively.

In the `Tcl` scripts, BMAP is emulated by assigning each batch (of X packets) to a stream, as well as an inter-stream gap (ISG) that approximates the inter-arrival time in the model. Starting from the first stream/batch, the next one is generated after the specified ISG, and so on. Then, the system loops back to the first stream/batch after the ISG of the last one, as shown in Fig. 6. Since the traffic generator allows up to 4096 streams when using Gigabit ports, and only 512 streams when using 10 Gigabit ports, we use the former to achieve better approximations of the batch size and inter-arrival time distributions.

B. Validation of Workload Profiling

As initially motivated in [6], the proposed approach seeks to profile VNF workloads beyond offered loads and utilization – specifically, to capture the workload burstiness, as characterized by the model parameters λ, $\beta_{(1)}$ and $\beta_{(2)}$.

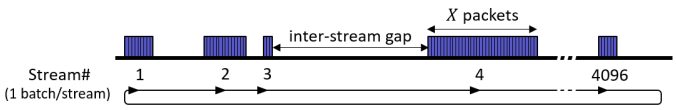


Fig. 6: Emulating BMAP arrivals.

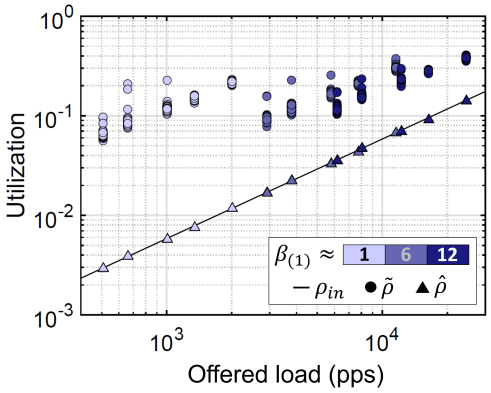


Fig. 7: Core utilization for varying VNF workload burstiness.

1) Offered load and utilization: Using the `vnstat` command, we obtain $\tilde{OL}$ with maximum and mean absolute percent error of 5.69% and 2.15%, respectively. Looking at Eq. (11), the same accuracy is expected for $\hat{\rho}$ with a constant value for μ.

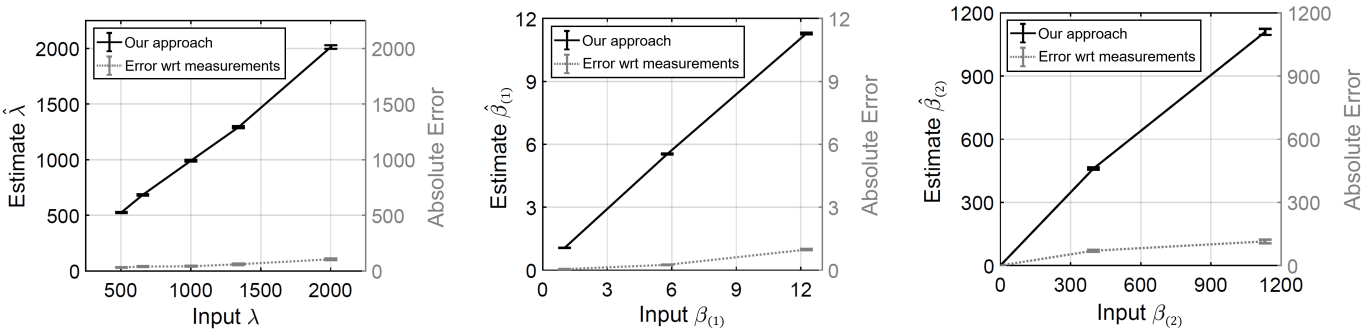
Fig. 7 shows a comparison between the measured ($\tilde{\rho}$) and estimated ($\hat{\rho}$) utilization values, with values computed from the input model parameters (ρ_{in} – the utilization due to actual packet processing). It can be observed how $\hat{\rho}$ fits with the model, while $\tilde{\rho}$ exhibits an overhead that seems to be highly correlated with the batch size; this further motivates the need to capture the traffic burstiness.

2) Burstiness: In this work, it was observed that with BMAP emulation the `idlestat` command gave reliable estimates for λ, which also comply with the theory of Poisson processes, while the software PMCs initially considered in [6] failed. Fig. 8a shows the estimates obtained based on Eq. (12), with $\alpha_1 = 0.910651$ and $\alpha_0 = -60.418339$. Tight confidence intervals and low absolute errors are achieved, even with varying values of $\beta_{(1)}$ aggregated in each test point.

A similar level of accuracy is expected with the $\beta_{(1)}$ estimates as they are solely based on $\tilde{OL}$ and $\hat{\lambda}$, as indicated in Eq. (13); the obtained estimates are shown in Fig. 8b. On the other hand, Fig. 8c shows the $\beta_{(2)}$ estimates obtained based on Eq. (14). Although the input and estimated values follow the same trend, a relatively higher variance (and hence, errors) are observed in $\hat{\beta}_{(2)}$ stemming from the fact that the starting point was the busy times (i.e., Eq. (15)), which also depend on other model parameters.

C. Validation of Network KPI Estimation

In this sub-section, we apply the results obtained from the workload profiling to the real-time estimation of networking KPIs – specifically, the VNF power consumption and system latency. Then, the estimates obtained from the power (i.e.,



(a) batch arrival rate (b) 1st factorial moment of X (c) 2nd factorial moment of X

Fig. 8: Estimating key statistical features of the workload burstiness.

Eq. (6)) and latency (i.e., Eq. (9)) models are compared with actual measurements.

As regards the power consumption, we suppose that the core power consumption due to the VNF can be obtained as $\tilde{\Phi} = \tilde{\Phi}_{cpkg} - \Delta\Phi$, where $\tilde{\Phi}_{cpkg}$ is the power consumed by the core part of the package (measurable with the `turbostat` command), and $\Delta\Phi$ is the overhead due to the other cores in the package. Recalling that the VNF is pinned to a core under (C_{1E}, P_{0T}), and the rest of the cores are in state C_6, we consider $\Delta\Phi \approx 15.93$ W in this work.

On the other hand, we suppose that the VNF latency can be obtained as $\tilde{D} = \tilde{D}_{ixia} - \Delta D$, where $\tilde{D}_{ixia}$ is the store-and-forward latency measurable from the user interface of the traffic generator, and ΔD is the overhead due to the 2-way transmission on a Gigabit link (i.e., $\approx 2\beta/1488095$) plus the busy overhead $\Delta T \approx 90$ μs (that includes τ_r, context switching (i.e., for which [25] proposed a rule of thumb of 30 μs), sleep transitions, etc.).

1) Power: Fig. 9a illustrates the behaviour of the power model for varying traffic burstiness. Intuitively, by looking at Eq. (6), the average core power consumption is linearly dependent on both λ and $\beta_{(1)}$ (embedded in ρ), although a stronger correlation is observed with the former.

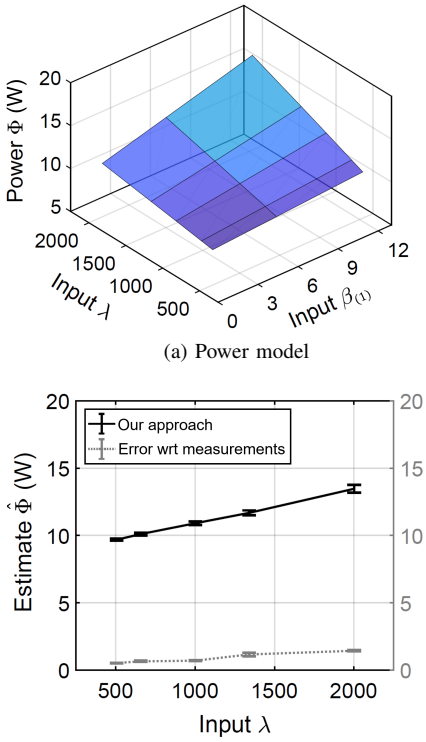
Results on the model-based power estimation, and its comparison with the actual measurements (in terms of absolute error) are shown in Fig. 9b. As before, tight confidence intervals and low absolute errors are achieved, even with varying values of $\beta_{(1)}$ aggregated in each test point.

2) Latency: Fig. 10a illustrates the behaviour of the latency model for varying traffic burstiness. Contrary to the power consumption, the average VNF latency incurred is more strongly linked to $\beta_{(1)}$ (and $\beta_{(2)}$) than to λ.

Results on the model-based latency estimation, and its comparison with the actual measurements (in terms of absolute error) are shown in Fig. 10b. Tight confidence intervals and low absolute errors are also achieved in this case, even with varying values of λ aggregated in each test point.

D. Validation on Facebook's Dataset

Finally, to further support our assumptions and exhaustive validation results with the wide range of (λ, β), we sample Facebook's web server cluster dataset [45] [46]. The timestamps corresponding to each (source) IP address are analyzed to derive the batch inter-arrival times and sizes.



(a) Power model

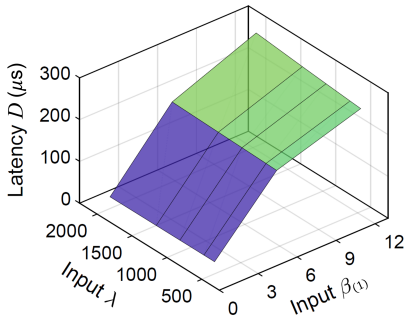
(b) Model-based estimates vs measurements

Fig. 9: Estimating the power consumption.

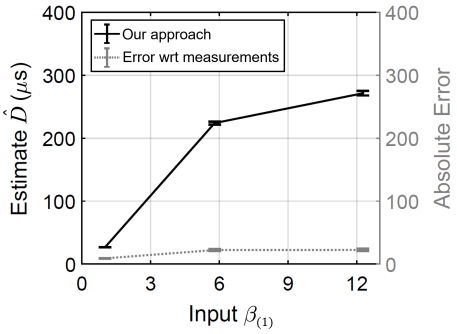
Looking at the top 500 addresses in the trace files (in terms of occurrences), Fig. 11a and 11b show the average batch inter-arrival times and sizes, respectively. A sort of steady-state phase can be observed from the 185th address (marked by the red dotted lines) – intuitively, this means that addresses in this subset have similar traffic burstiness and hence, comparable system behaviors. With this in mind, 5 IP addresses are randomly chosen from the said subset for the detailed validation.

In the following, the considered batch inter-arrival times and sizes are first fitted to exponential and generalized Pareto distributions, respectively. Then, considering the traffic of each address as input to the SUT, the network KPI values obtained with the proposed model-based estimation are compared with actual measurements.

1) Distribution fitting: Using Matlab's `fitdist` function, we obtain the distribution parameters resulting from fitting the input distributions with the considered models. The

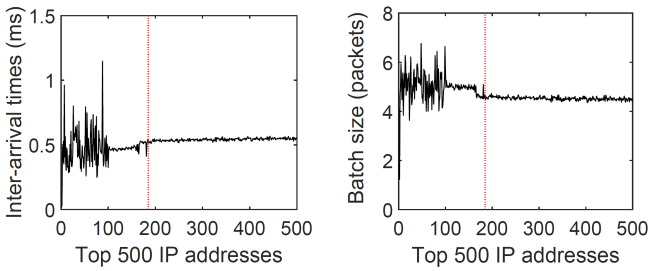


(a) Latency model



(b) Model-based estimates vs measurements

Fig. 10: Estimating the system latency.



(a) Inter-arrival times (b) Batch sizes

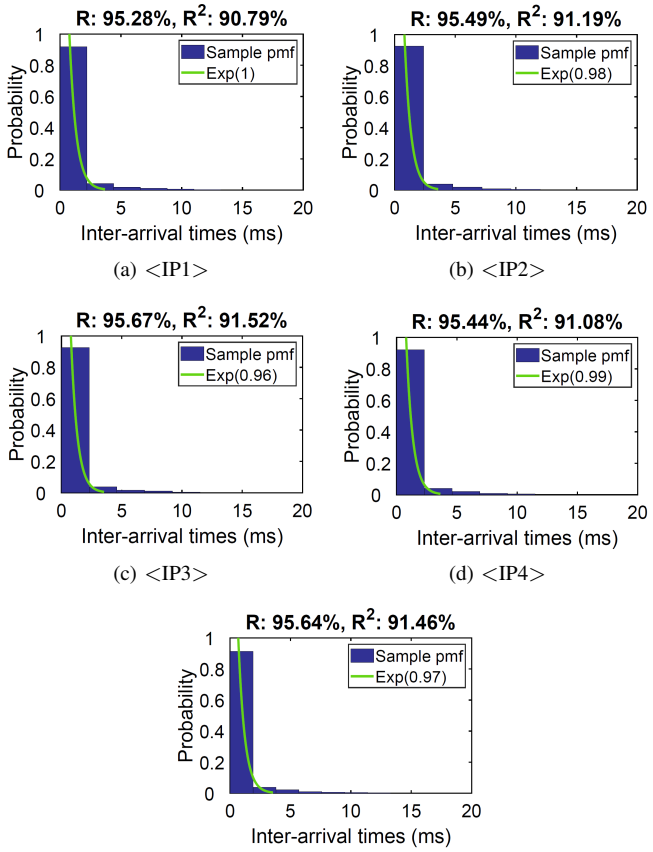
Fig. 11: Average inter-arrival times and batch sizes for the top 500 IP addresses by occurrence.

goodness of fit is then measured in terms of the coefficients of *correlation* (**R**) and *determination* (**R²**).

Figs. 12 and 13 show how well the distributions fit for the 5 IP addresses. Particularly, it can be observed in Fig. 12 that the input batch inter-arrival times have $\mathbf{R} > 95\%$ and $\mathbf{R^2} > 90\%$ with the exponential distribution, while the batch sizes in Fig. 13 have **R** and $\mathbf{R^2}$ values over 99% with the generalized Pareto distribution. Such high values of **R** and $\mathbf{R^2}$ confirm that the samples considered from Facebook's dataset are well-represented by the models.

2) Network KPI testing: The traffic of each address is fed as input to the SUT in order to evaluate the corresponding power consumptions and latencies. As in Sub-section VI-C, the values obtained with the proposed model-based estimation and actual measurements are compared in terms of absolute errors.

Fig. 14a shows the average power consumption for the 5 IP addresses, while Fig. 14b the average system latencies. Tight confidence intervals and low absolute errors (i.e., $\approx 3\%$ for power and $\approx 6\%$ for latency) can be observed for both



(a) <IP1> (b) <IP2>

(c) <IP3> (d) <IP4>

(e) <IP5>

Fig. 12: Fitting batch inter-arrival times to exponential distributions, Exp(<mean>).

KPIs across all cases. Such accuracies for real-time network KPI estimation demonstrate how the proposed approach can be a powerful tool towards achieving the required scalability and sustainability levels in next-generation network/service management and orchestration.

VII. Conclusion

NFV is an emerging softwarization solution that brings flexibility and programmability through the software implementation of network functions (i.e., as VNFs) on COTS hardware. A number of issues surround the performance and energy efficiency of such virtual implementations, with respect to their physical counterparts. This work seeks to facilitate scalable and sustainable network/service management and orchestration mechanisms, through a novel model-based analytics approach for profiling VNF workloads, towards real-time estimation of network KPIs.

Particularly, the M^X/G/1/SET core model is considered to capture both the workload burstiness and system setup times. A complete analytical characterization of the system is presented, based on which the model-based analytics approach is built upon. Key model parameters are exposed from available and easily measurable PMCs in Linux host servers. In terms of generalizability, the proposed approach goes beyond current trends in ML-based analytics, where models are tightly coupled with the training data.

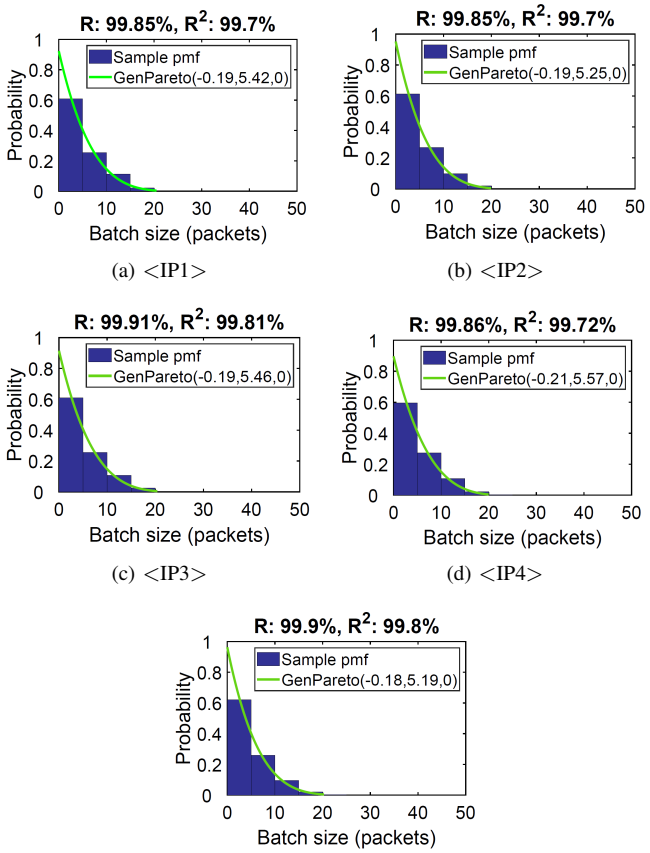


Fig. 13: Fitting batch sizes to generalized Pareto distributions, GenPareto(<shape>,<scale>,<location>).

Experimental evaluations have been performed on a SUT equipped with Intel® Xeon® E5-2643 v3 3.40GHz processors, with input traffic generated to emulate BMAP arrivals through scripting in an Ixia NX2 traffic generator, as well as some samples from Facebook's web server cluster traces. Results show good estimation accuracies for both VNF workload profiling and network KPI estimation, with respect to the input traffic and actual measurements, respectively. This demonstrates how the proposed approach can be a powerful tool, not only for augmenting the capabilities of an NFVI's VIM, but also in the development of next-generation resource/service provisioning solutions.

Appendix A
Busy Period Analysis

We adopt the approach presented in [47], decomposing the busy period B of an M^X/G/1/SET queue into two components: (a) the initial busy period B_τ – in which all the customers that arrived during the setup SET are served, and (b) the ordinary busy period B_X that corresponds to the busy period of an M^X/G/1 queue – in which the batch initiating the setup and the rest that arrived while the core is busy are served.

Considering that the busy period density $B(t)$ is given by the convolution of the probability densities $B_\tau(t)$ and $B_X(t)$, then its Laplace transform is simply obtained as the product $B^*(\theta) = B^*_\tau(\theta)B^*_X(\theta)$.

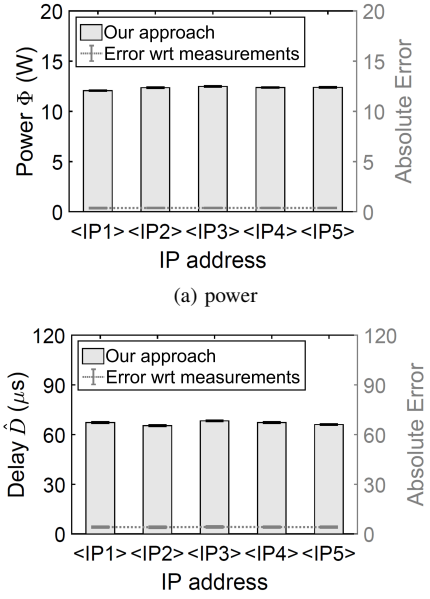


Fig. 14: Estimating network KPIs with Facebook's dataset.

Let the random variable η_τ denote the number of batch arrivals during SET. Given that $SET = t$ and $\eta_\tau = m$, then B_τ is distributed as the sum of the lengths of m independent ordinary busy periods $B_{X1}, \ldots, B_{Xm}$ [33]. By first conditioning on SET and η_τ, and then averaging out, we obtain the Laplace transform of $B_\tau(t)$ as:

$$\begin{aligned} B^*_\tau(\theta) &= \int_0^\infty \sum_{m=1}^{\infty} e^{-\lambda t}\frac{(\lambda t)^m}{m!} E\{e^{-\theta[B_{X1}+\ldots+B_{Xm}]}\}\tau(t)\ dt \\ &= \tau^*\big(\lambda - \lambda B^*_X(\theta)\big) \end{aligned} \tag{A.1}$$

Similarly, let the random variables S_{X_1}, X_1 and η_1 denote the service time of the initiating batch, the number of customers in this batch, and the number of batch arrivals during S_{X_1}, respectively. Given that $S_{X_1} = t$, $X_1 = j$ and $\eta_1 = n$, then in the same way as before, B_X is distributed as the sum of the lengths of t and n independent ordinary busy periods. By conditioning on S_{X_1}, X_1 and η_1, and proceeding as before, we obtain the Laplace transform of $B_X(t)$ as:

$$\begin{aligned} B^*_X(\theta) &= \sum_{j=1}^{\infty}\beta_j \int_0^\infty \sum_{n=1}^{\infty} e^{-\lambda t}\frac{(\lambda t)^n}{n!} E\{e^{-\theta[t+B_{X1}+\ldots+B_{Xn}]}\} \\ &\quad \cdot s_1(t) * \cdots * s_j(t)\ dt \\ &= \sum_{j=1}^{\infty}\beta_j s^*\big(\theta+\lambda-\lambda B^*_X(\theta)\big)^j \\ &= X\big(s^*\big(\theta+\lambda-\lambda B^*_X(\theta)\big)\big) \end{aligned} \tag{A.2}$$

Appendix B
System State Probabilities

The Probability Generating Function (PGF) $P(z)$ of the number of customers in an M^X/G/1/SET queueing system at a random epoch can be expressed as the product $P(z) = P_\tau(z)P_X(z)$, with $P_\tau(z)$ being the PGF of the

number of customer arrivals during the residual life of the vacation period (i.e., $I + SET$) and

$$P_X(z) = \frac{(1-\rho)(1-z)s^*\big(\lambda - \lambda X(z)\big)}{s^*\big(\lambda - \lambda X(z)\big) - z} \tag{B.1}$$

the well-known PGF of the number of customers in the ordinary M^X/G/1 system.

Let $\varphi(z) = X(z)\tau^*\big(\lambda - \lambda X(z)\big)$ be the PGF of the number of customer arrivals during a generic vacation period, from which we obtain the average number of customers by the end of SET as $\varphi_{(1)} = \frac{d\varphi(z)}{dz}|_{z=1} = \beta_{(1)}(1 + \lambda\tau_{(1)})$. By simplifying the general expression found in [48], we arrive to

$$P_\tau(z) = \frac{1 - X(z)\tau^*\big(\lambda - \lambda X(z)\big)}{(1 + \lambda\tau_{(1)})\big(1 - X(z)\big)} \tag{B.2}$$

ACKNOWLEDGMENT

This work was supported by the European Commission in the framework of the H2020 5G-PPP MATILDA Project (contract no. 761898).

REFERENCES

[1] M. Chiosi et al., "Network Functions Virtualization: An Introduction, Benefits, Enablers, Challenges & Call For Action," White Paper, 2012. [Online]. Available: http://portal.etsi.org/NFV/NFV_White_Paper.pdf

[2] "Advanced Configuration and Power Interface Specification." [Online]. Available: http://www.acpi.info/DOWNLOADS/ACPIspec40a.pdf

[3] R. Bolla, R. Bruschi, F. Davoli, and F. Cucchietti, "Energy Efficiency in the Future Internet: A Survey of Existing Approaches and Trends in Energy-Aware Fixed Network Infrastructures," *IEEE Commun. Surveys Tuts.*, vol. 13, no. 15, pp. 223–244, 2011.

[4] E. Hernandez-Valencia, S. Izzo, and B. Polonsky, "How Will NFV/SDN Transform Service Provider OPEX?" *IEEE Netw.*, vol. 29, no. 3, pp. 60–67, May 2015.

[5] "Network Functions Virtualisation (NFV); Management and Orchestration," ETSI NFV ISG Specification, 2014. [Online]. Available: http://www.etsi.org/deliver/etsi_gs/NFV-MAN/001_099/001/01.01.01_60/gs_nfv-man001v010101p.pdf

[6] R. Bruschi, F. Davoli, P. Lago, and J. F. Pajo, "Model-based Analytics for Profiling Workloads in Virtual Network Functions," in *Proc. 2017 IEEE Int. Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, Atlanta, GA, USA, May 2017, pp. 916–921.

[7] G. Choudhury, "An M^X/G/1 Queueing System with a Setup Period and a Vacation Period," *Queueing Syst.*, vol. 36, no. 1-3, pp. 23–38, 2000.

[8] R. Bolla, R. Bruschi, A. Carrega, and F. Davoli, "Green Networking with Packet Processing Engines: Modeling and Optimization," *IEEE/ACM Trans. Netw.*, vol. 22, no. 1, pp. 110–123, Feb. 2014.

[9] R. Bolla, R. Bruschi, A. Carrega, F. Davoli, and J. F. Pajo, "Corrections to: "Green Networking With Packet Processing Engines: Modeling and Optimization"," *IEEE/ACM Trans. Netw.*, 2018.

[10] L. Duan, D. Zhan, and J. Hohnerlein, "Optimizing Cloud Data Center Energy Efficiency via Dynamic Prediction of CPU Idle Intervals," in *Proc. 8th IEEE Int. Conf. Cloud Comput. (CLOUD)*, New York, NY, USA, Jun. 2015, pp. 985–988.

[11] "Network Functions Virtualisation (NFV); Virtualisation Technologies; Report on the application of Different Virtualisation Technologies in the NFV Framework," ETSI NFV ISG Specification, 2016. [Online]. Available: https://www.etsi.org/deliver/etsi_gs/NFV-EVE/001_099/004/01.01.01_60/gs_nfv-eve004v010101p.pdf

[12] T. Barbette, C. Soldani, and L. Mathy, "Fast Userspace Packet Processing," in *Proc. 11th ACM/IEEE Symp. Archit. Netw. Commun. Syst. (ANCS)*, Oakland, CA, USA, May 2015, pp. 5–16.

[13] P. Kutch and B. Johnson, "SR-IOV for NFV Solutions: Practical Considerations and Thoughts," Feb. 2017. [Online]. Available: https://www.intel.com/content/dam/www/public/us/en/documents/technology-briefs/sr-iov-nfv-tech-brief.pdf

[14] M. Kourtis et al., "Enhancing VNF Performance by Exploiting SR-IOV and DPDK Packet Processing Acceleration," in *Proc. 2015 IEEE Conf. Netw. Function Virtualization Softw. Defined Netw. (NFV-SDN)*, San Francisco, CA, USA, Nov. 2015, pp. 49–59.

[15] J. Li, S. Xue, W. Zhang, R. Ma, Z. Qi, and H. Guan, "When I/O Interrupt Becomes System Bottleneck: Efficiency and Scalability Enhancement for SR-IOV Network Virtualization," *IEEE Trans. Cloud Comput.*, Dec. 2018.

[16] P. Li, X. Wu, Y. Ran, and Y. Luo, "Designing Virtual Network Functions for 100 GbE Network Using Multicore Processors," in *Proc. 2017 ACM/IEEE Symp. Archit. Netw. Commun. Syst. (ANCS)*, Beijing, China, May 2017, pp. 49–59.

[17] C. Möbius, W. Dargie, and A. Schill, "Power Consumption Estimation Models for Processors, Virtual Machines, and Servers," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 6, pp. 1600–1614, Jun. 2014.

[18] M. Dayarathna, Y. Wen, and R. Fan, "Data Center Energy Consumption Modeling: A Survey," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 1, pp. 732–794, 1Q 2016.

[19] M. Dimitrov, K. Doshi, R. Khanna, K. Kumar, and C. Le, "Coordinated Optimization: Dynamic Energy Allocation in Enterprise Workload," *Intel® Technol. J.*, vol. 16, no. 2, pp. 32–51, 2012.

[20] "Volume 3B: System Programming Guide, Part 2," Intel® 64 and IA-32 Architectures Software Developer's Manual, 2016. [Online]. Available: https://www.intel.com/content/dam/www/public/us/en/documents/manuals/64-ia-32-architectures-software-developer-vol-3b-part-2-manual.pdf

[21] J. Prados-Garzon, J. J. Ramos-Munoz, P. Ameigeiras, P. Andres-Maldonado, and J. M. Lopez-Soler, "Latency Evaluation of a Virtualized MME," in *Proc. 2015 Wireless Days (WD)*, Toulouse, France, Mar. 2016.

[22] W. Chiang and J. Wen, "Design and Experiment of NFV-Based Virtualized IP Multimedia Subsystem," in *Proc. 3rd Int. Conf. Comput. Commun. Syst. (ICCCS)*, Nagoya, Japan, Apr. 2018, pp. 397–401.

[23] M. S. Yoon and A. E. Kamal, "NFV Resource Allocation Using Mixed Queuing Network Model," in *Proc. 2016 IEEE Global Commun. Conf. (GLOBECOM)*, Washington, DC, USA, Dec. 2016.

[24] F. C. Chua, J. Ward, Y. Zhang, P. Sharma, and B. A. Huberman, "Stringer: Balancing Latency and Resource Usage in Service Function Chain Provisioning," *IEEE Internet Comput.*, vol. 20, no. 6, pp. 22–31, Nov. 2016.

[25] S. Gebert, T. Zinner, S. Lange, C. Schwartz, and P. Tran-Gia, "Performance Modeling of Softwarized Network Functions Using Discrete-Time Analysis," in *Proc. 28th Int. Teletraffic Congr. (ITC28)*, vol. 1, Würzburg, Germany, Sep. 2016, pp. 234–242.

[26] R. Bruschi, F. Davoli, P. Lago, and J. F. Pajo, "Joint Power Scaling of Processing Resources and Consolidation of Virtual Network Functions," in *Proc. 5th IEEE Int. Conf. Cloud Netw. (CloudNet)*, Pisa, Italy, Oct. 2016, pp. 70–75.

[27] R. Bolla, R. Bruschi, and P. Lago, "The Hidden Cost of Network Low Power Idle," in *Proc. 2013 IEEE Int. Conf. Commun. (ICC)*, Budapest, Hungary, Jun. 2013, pp. 4148–4153.

[28] R. Schöne, D. Molka, and M. Werner, "Wake-up Latencies for Processor Idle States on Current x86 Processors," *Comput. Sci. - Res. Develop.*, vol. 30, no. 2, pp. 219–227, 2015.

[29] "KVM." [Online]. Available: http://www.linux-kvm.org/

[30] S. Zeng and Q. Hao, "Network I/O Path Analysis in the Kernel-Based Virtual Machine Environment Through Tracing," in *Proc. 1st Int. Conf. Inform. Sci. Eng. (ICISE)*, Nanjing, China, Dec. 2009, pp. 2658–2661.

[31] C. Xu, Z. Zhao, H. Wang, R. Shea, and J. Liu, "Energy Efficiency of Cloud Virtual Machines: From Traffic Pattern and CPU Affinity Perspectives," *IEEE Syst. J.*, vol. 11, pp. 835–845, Jun. 2017.

[32] "Ethtool(8) - Linux Man Page." [Online]. Available: https://linux.die.net/man/8/ethtool

[33] H. Tijms, *A First Course in Stochastic Models*. John Wiley & Sons Ltd, England, 2003.

[34] A. Klemm, C. Lindemann, and M. Lohmann, "Modeling IP Traffic using the Batch Markovian Arrival Process," *Perform. Eval.*, vol. 54, pp. 149–173, Oct. 2003.

[35] P. Salvador, A. Pacheco, and R. Valadas, "Modeling IP Traffic: Joint Characterization of Packet Arrivals and Packet Sizes using BMAPs," *Comput. Networks*, vol. 44, pp. 335–352, Feb. 2004.

[36] D. Lucantoni, "The BMAP/G/1 Queue: A Tutorial," in *Perform. Eval. Comput. Commun. Syst.*, L. Donatiello and R. Nelson, Ed. Springer Berlin Heidelberg, 1993, pp. 330–358.

[37] E. Hyytiä, R. Righter, J. Virtamo, and L. Viitasaari, "Value (Generating) Functions for the M^X/G/1 Queue," in *Proc. 29th Int. Teletraffic Congr. (ITC29)*, vol. 1, Genoa, Italy, Sep. 2017, pp. 232–240.

[38] "Idlestat - A CPU Power-state Analysis Tool." [Online]. Available: http://manpages.ubuntu.com/manpages/xenial/man1/idlestat.1.html

[39] "vnStat - A Console-based Network Traffic Monitor." [Online]. Available: http://manpages.ubuntu.com/manpages/bionic/man1/vnstat.1.html

[40] P. W. Glynn and W. Whitt, "Estimating the Asymptotic Variance with Batch Means," *Oper. Res. Lett.*, vol. 10, no. 8, pp. 431–435, Nov. 1991.

[41] "OpenWrt." [Online]. Available: https://openwrt.org/
[42] R. Bolla, R. Bruschi, O. M. J. Ortiz, and P. Lago, "An Experimental Evaluation of the TCP Energy Consumption," *IEEE J. Sel. Areas Commun.*, vol. 33, no. 12, pp. 2761–2773, Dec. 2015.
[43] "Turbostat - Report Processor Frequency and Idle Statistics." [Online]. Available: http://manpages.ubuntu.com/manpages/xenial/man8/turbostat.8.html
[44] D. Hackenberg, T. Ilsche, R. Schöne, D. Molka, M. Schmidt, and W. E. Nagel, "Power Measurement Techniques on Standard Compute Nodes: A Quantitative Comparison," in *Proc. 2013 IEEE Int. Symp. Perform. Anal. Syst. Softw.*, Austin, TX, USA, Apr. 2013, pp. 194–204.
[45] A. Roy, H. Zeng, J. Bagga, G. Porter, and A. C. Snoeren, "Inside the Social Network's (Datacenter) Network," in *Proc. 2015 ACM Conf. Special Interest Group Data Commun. (SIGCOMM)*, London, United Kingdom, Aug. 2015, pp. 123–137.
[46] Facebook, "FBFlow Dataset." [Online]. Available: https://www.facebook.com/network-analytics
[47] P. Mevert, "The Alternating Queueing Process with Setup Times," Case Institute of Technology, Tech. Memo. 63, Jun. 1966.
[48] J.-C. Ke, "Batch Arrival Queues Under Vacation Policies With Server Breakdowns and Startup/Closedown Times," *Appl. Math. Model.*, vol. 31, no. 7, pp. 1282–1292, 2007.