An Analytical Model to Design Processor Sharing for SDN/NFV Nodes

Giuseppe Faraci, Alfio Lombardo, Giovanni Schembra
Dipartimento di Ingegneria Elettrica, Elettronica e Informatica (DIEEI) - University of Catania
{giuseppe.faraci, lombardo, schembra}@dieei.unict.it

Abstract— **The introduction of the two new paradigms Software Defined Networks (SDN) and Network Functions Virtualization (NFV) in the Internet is revolutionizing its design and management, creating many new challenging problems regarding resource allocation and orchestration. Little attention has been paid up to now to the "intranode" resource allocation. In this context, this paper introduces an analytical model of an SDN/NFV node that implements NARR, a processor sharing strategy proposed by the same authors to reduce delays by assigning a larger processor portion to the flows that will leave the node through a less loaded output network interface card (NIC). Numerical results analytically demonstrates the gain achieved by applying NARR, and give important insights for future work.**

Keywords-component; SDN; NFV; Processor sharing; Markov model; QoS

I. INTRODUCTION

In the last few years, SDN and NFV have been proposed as two revolutionary paradigms to transform the ossified Internet into a flexible and programmable network [1].

Software Defined Network (SDN) [2] aims at decoupling the network-control plane from the network-data plane, also providing programmatic interfaces into network equipment. This technology, initially applied to simplify building and managing large IP/Ethernet networks by exercising central control over forwarding, and to support communications in virtual and overlay networks, is widely used today in different contexts like for examples data center LANs and the Google's IP core network.

Network functions virtualization (NFV) [3], on the other hand, is a carrier-driven initiative to virtualize network functions and network services by using technologies inherited by the world of data centers and clouds. The main objective of NFV is to improve service flexibility by using a more agile software-based framework for building service features, consequently reducing both CAPEX (CAPital EXpenditure) and OPEX (OPerating EXpenditure) of Telco Operator networks. According to a cloud perspective, adoption of network overlays for virtual function provisioning makes NFV the largest consumer of cloud networking and SDN services.

The application of both the paradigms is introducing new challenges in the design and management of all the components of a telecommunications network. Specific problems are arising concerning orchestration and resource allocation, and classical solutions coming from traditional networking scenarios and cloud computing are not able to solve them, mainly due to the intrinsic characteristic of an SDN/NFV network that can be considered as a big distributed data center where the internal delay is not negligible [4-5].

The management problem of an SDN/NFV network can be addressed at two different levels: internode and intranode. The first approach, which has been considered for example in [6], aims at optimizing placement and consolidation of Virtual Network Functions (VNFs), and deciding traffic routing in such a way that each flow crosses the nodes where the assigned instances of the required VNFs are running. On the contrary, very little has been done in the past literature regarding the intranode approach, consisting in optimizing performance within an SDN/NFV node. In this regard, the authors in [7] have proposed Network-Aware Round Robin (NARR), a processor-sharing technique that is able to overcome classical processor-sharing techniques [8] by leveraging the correlation between the behavior of the queues containing packets waiting for processor service and the behavior of the queues associated to the network interface cards (NICs). The authors started from the consideration that packets that have received the service of a network function from a virtual machine (VM) running on a given node will be enqueued to wait for transmission through an output NIC. Consequently, the NARR strategy dynamically changes the portions of the CPU assigned to each VNF according to the state of the output NIC queues, giving a larger CPU portion to serve packets that will leave the node through the NIC that is currently less loaded, with the aim of minimizing wastes of the NIC output link capacities.

Starting from this point, this paper defines a discrete-time Markov multidimensional chain that models the behavior of an SDN/NFV node that implements the NARR strategy, with the aim of supporting both the designer in deploying network nodes, and the orchestrator in its management role.

The rest of this paper is organized as follows. Section II will provide a description of the considered SDN/NFV node and of the NARR strategy. Section III will introduce the proposed analytical model of the system. Section IV will present some numerical results. Finally, Section V will draw some conclusions and describe some ideas of future work.

II. SYSTEM DESCRIPTION

The system we consider in the rest of the paper is an SDN/NFV node as described in [4-5]. Its block diagram is sketched in Fig. 1. According to the Management and Orchestration (MANO) ETSI specification [9] and the NFV Infrastructure ETSI specification [10] documents, it represents the Compute and the Network domains of an NFV Point of Presence (PoP) node.

Its main components are the *Processor*, indicated in Fig. 1 as CPU, and the NICs. A set of queues is associated to the Processor and the NICs aimed at buffering packets waiting for service by both of them. Each VNF is run on the Processor in a separate virtual machine (VM).

Let M be the number of VNFs that are running in the node, and L the number of output NICs ($M = 2$ and $L = 3$ in Fig. 1). Let $K^{(NIC)}$ be the size of the queue associated to each NIC, that is, the maximum number of packets that each queue can contain, and $\mu^{(NIC)}$ be the transmission rate of the output link associated to each NIC, expressed in bit/s.

Two blocks control the behavior of the node: the Flow Distributor and the Processor Arbiter. The *Flow Distributor* block is an SDN-compliant element that has the task of routing each entering flow through the required VNF, according to the

control messages received by the remote SDN Controller residing in the Orchestrator. The most common protocol that can be used for the communications between the SDN Controller and the Flow Distributor block is OpenFlow. The *Processor Arbiter* is in charge of dynamically assigning portions of processor rate to each inside-function queue.

Let $\mu^{(P)}$ be the total Processor rate, expressed in packets/s. This rate is shared among all the active VMs according to a processor sharing strategy. Here we consider the NARR strategy proposed in [7], briefly described below for the sake of completeness. Let $\underline{\mu}^{(F)}$ be the array whose generic element, $\mu_{[m]}^{(F)}$, with $m \in \{1,\ldots,M\}$, is the portion of the Processor rate assigned to the VM running the VNF F_m. Of course, we have:

$$\sum_{m=1}^{M} \mu_{[m]}^{(F)} = \mu^{(P)} \tag{1}$$

Once a packet has been served by the processor to receive the required VNF, it is enqueued in one of the queues associated to the output NICs, according to the path decided by the remote *Orchestrator*. We will indicate the queue associated to the generic NIC l as $Q_l^{(NIC)}$.

In accordance with the NARR strategy, the block relative to each VNF is constituted by a set of L parallel queues, in the following referred to as *inside-function queues*. The generic l-th inside-function queue of the VNF F_m, indicated as $Q_{m,l}$ in Fig. 1, is used to enqueue packets that, after receiving the service of the VNF F_m, will leave the node through the NIC l. Let $K_{Ins}^{(F)}$ be the size of each inside-function queue. Each inside-function queue of the generic VNF F_m receives a portion of the processor rate assigned to that VNF. Let $\mu_{[m,l]}^{(F_{Ins})}$ be the portion of the processor rate assigned to the queue $Q_{m,l}$ of the VNF F_m. Of course, we have:

$$\sum_{l=1}^{L} \mu_{[m,l]}^{(F_{Ins})} = \mu_{[m]}^{(F)} \tag{2}$$

The portion of processor rate associated to each inside-function queue is dynamically changed by the *Processor Arbiter* according to the NARR strategy. The NARR (Network-Aware Round-Robin) processor-sharing strategy works observing the state of both the inside-function queues and the NIC queues, with the goal of reducing the inactivity periods of the NIC output links as far as possible. It leverages the fact that packets that have received the service of a VNF are enqueued to wait for transmission through a given NIC. So, in order to avoid output link capacity waste, NARR dynamically changes the portions of the CPU assigned to each VNF, and then to each inside-function queue, according to the state of the output NIC queues. More specifically, larger CPU portions are given to serve queues of packets that will leave the node through less-loaded NICs.

More in deep, the Processor Arbiter decides the processor rate portions according to two different steps. The first step regards the assignment of the processor-rate portion to the aggregation of queues whose output is a specific NIC. To this purpose, let us consider a virtual queue that contains all the

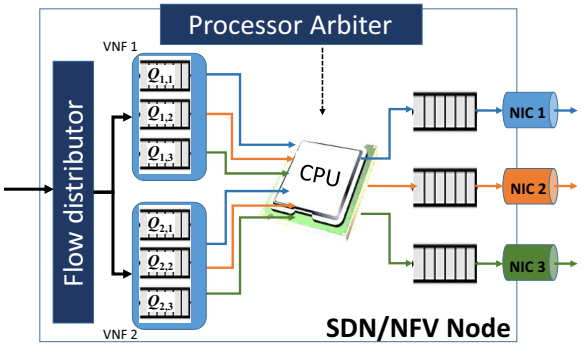


Figure 1: SDN/NFV Node Block Diagram

packets of all the inside-function queues $Q_{m,l}$, for each $m \in [1,M]$, that is all the packets that will leave the node through the NIC l. Let us indicate this virtual queue as $Q_{Aggr}^{(\rightarrow NIC_l)}$, and its service rate as $\mu_{Aggr}^{(\rightarrow NIC_l)}$. According to the particular case shown in Fig. 1, for example $Q_{Aggr}^{(\rightarrow NIC_1)}$ is the aggregate of the inside-function queues $Q_{1,1}$ and $Q_{2,1}$. Of course, we have:

$$\mu_{Aggr}^{(\rightarrow NIC_l)} = \sum_{m=1}^{M} \mu_{[m,l]}^{(F_{Ins})} \qquad \forall l \in \{1,\ldots,L\} \tag{3}$$

Taking into account the objective of privileging the flows that will leave the node through underloaded NICs, the Processor Arbiter calculates $\mu_{Aggr}^{(\rightarrow NIC_l)}$ by considering each NIC queue as divided in ranges, in order to assign the same processor portion to the queue lengths belonging to the same range. If we indicate the number of ranges of a NIC queue as N_2, the interval size is $\Delta_2 = K^{(NIC)}/N_2$. Given a current state of a NIC queue, x, we will indicate the relative queue range as $\varphi_2(x)$. It can be easily derived as $\varphi_2(x) = \lceil x/\Delta_2 \rceil$, where the operator $\lceil \gamma \rceil$ represents the minimum integer not less than γ.

Now, let us define a term q_{ref} as a reference target value calculated from the state of the NIC queue that has the highest length, amplified with a coefficient $\vartheta \geq 1$, and truncated to the maximum queue size $K^{(NIC)}$. If we indicate the state of the generic j-th NIC queue (i.e. the number of packets in the queue) as $S_j^{(NIC)}$, we have:

$$q_{ref} = \min\left\{\vartheta \cdot \max_j\left(S_j^{(NIC)}\right),\ K^{(NIC)}\right\} \tag{4}$$

The meaning of ϑ is that, for low values of it approaching the unit, the most loaded queues are not overloaded by the inside-function queues because the service rate of them is maintained very low; the other queues receive packets with a rate that is proportional to the distance between the range of their length and the range of q_{ref}. After an extensive set of simulations, we noticed that choosing $\vartheta = 1$ implies bad performance because there is always a group of functions that are not served. Instead, choosing ϑ in the interval $]1,2]$ gives

almost equivalent performance. For this reason, in our system we have decided to use $\vartheta = 1.2$. If we indicate the queue range of the reference target value q_{ref} as φ_{ref}, and the queue interval corresponding to the state of the generic *j*-th NIC as $\varphi_2\left(S_l^{(NIC)}\right)$, we have:

$$\varphi_{ref} = \begin{cases} \varphi_2(q_{ref}) & \text{if } \varphi_2(q_{ref}) > \varphi_2\left(\max_j\left(S_j^{(NIC)}\right)\right) \\ \min\left\{\begin{matrix}\varphi_2(q_{ref})+1 \\ N_2 \cdot L\end{matrix}\right\} & \text{if } \varphi_2(q_{ref}) = \varphi_2\left(\max_j\left(S_j^{(NIC)}\right)\right) \end{cases} \quad (5)$$

The Processor Arbiter assigns a processor portion $\mu_{Aggr}^{(\rightarrow NIC_l)}$ to the $Q_{Aggr}^{(\rightarrow NIC_l)}$ queue that is proportional to the difference between the q_{ref} queue interval and the queue interval of the $Q_{Aggr}^{(\rightarrow NIC_l)}$ queue, that is:

$$\mu_{Aggr}^{(\rightarrow NIC_l)} = \frac{\varphi_{ref} - \varphi_2\left(S_l^{(NIC)}\right)}{\sum_{j=1}^{L}\left\{\varphi_2\left(q_{ref}\right) - \varphi_2\left(S_j^{(NIC)}\right)\right\}} \mu^{(P)} \quad (6)$$

The second assignment step decides the processor portion for each inside-function queue. Let us consider the generic *l*-th inside-function queue of the function F_m, i.e. the queue $Q_{m,l}$. Its service rate is calculated as proportional to the current state of this queue in comparison with the other *l*-th queues of the other functions. To this purpose, let us indicate the state of the virtual queue $Q_{Aggr}^{(\rightarrow NIC_l)}$ as $S_{Aggr}^{(\rightarrow NIC_l)}$. Of course, it can be calculated as the sum of the states of all the inside-function queues $Q_{m,l}$, for each $m \in [1, M]$, that is:

$$S_{Aggr}^{(\rightarrow NIC_l)} = \sum_{k=1}^{M} S_{k,l}^{(F_{ins})} \quad (7)$$

So, the service rate of the inside-function queue $Q_{m,l}$ is determined as a fraction of the service rate assigned at the first step to the virtual queue $Q_{Aggr}^{(\rightarrow NIC_l)}$, $\mu_{Aggr}^{(\rightarrow NIC_l)}$, as follows:

$$\mu_{[m,l]}^{(F_{ins})} = \frac{\varphi_1\left(S_{m,l}^{(F_{ins})}\right)}{\varphi_1\left(S_{Aggr}^{(\rightarrow NIC_l)}\right)} \mu_{Aggr}^{(\rightarrow NIC_l)} \quad (8)$$

where, similarly to the first step, $\varphi_1(x)$ is the range related to an inside-function queue length *x* when the queue is divided in N_1 ranges of size Δ_1. Of course, if at any time an inside-function queue remains empty, the processor rate portion assigned to it will be shared to the other queues proportionally to the processor portions previously assigned. Likewise, if at some instant an empty queue receives a new packet, the previous processor rate portion is re-assigned to that queue.

III. The Analytical Model

In this section we introduce the analytical model of the system described so far. To this purpose, as shown in Fig. 2, we concentrate our focus on the generic inside-function queue $Q_{m,l}$, here referred to as $Q1$, and the generic NIC $Q_l^{(NIC)}$, here referred to as $Q2$.

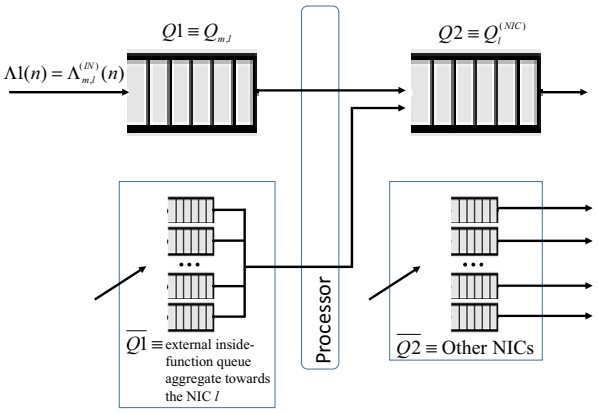


Figure 2: Analytical Representation of an SDN/NFV Node

First, let us describe the arrival process of the queue Q1, and the service process of the queue Q2. To this purpose, let $\Lambda_{m,l}^{(IN1)}(n)$, in the following also indicated as $\Lambda 1(n)$ to be coherent with the notation in Fig. 2, be the switched batch Bernoulli Process (SBBP) characterizing the input traffic of the $Q1$ queue, that is the aggregate of flows that require the function F_m and leave the node through the NIC *l*. According to the SBBP model definition, $\Lambda 1(n)$ is a discrete-time Markov-modulated process characterized by the set $\left\{P^{(IN1)}, B^{(IN1)}, \Im^{(IN1)}, \Psi^{(IN1)}\right\}$, where:

- $P^{(IN1)}$ is the transition probability matrix of the underlying Markov chain of $\Lambda 1(n)$. If we describe this chain with the discrete-time process $S^{(IN1)}(n)$, the generic element of $P^{(IN1)}$ is:
$$P_{[s'_{IN1}, s''_{IN1}]}^{(IN1)} = \Pr\left\{S^{(IN1)}(n+1) = s''_{IN1} \middle| S^{(IN1)}(n) = s'_{IN1}\right\} \quad (9)$$
- $B^{(IN1)}$ is the arrival probability matrix describing the arrival process for each state of the underlying Markov chain of $\Lambda 1(n)$. Its generic element is defined as follows:
$$B_{[s'_{IN1}, \alpha_1]}^{(IN1)} = \Pr\left\{\Lambda^{(IN1)}(n) = \alpha_1 \middle| S^{(IN1)}(n) = s'_{IN1}\right\} \quad (10)$$
- $\Im^{(IN1)}$ is the state space of the underlying Markov chain $S^{(IN1)}(n)$;
- $\Psi^{(IN1)}$ is the set of possible values that the process $\Lambda 1(n)$ can assume, that is, the state space of the number of packets that can arrive to the node in one slot.

In order to describe the service process of the NIC queue Q2, let us consider that packets buffered in it are served by a link with a constant bit rate. Consequently, according to the packet size, we describe the Q2 service process, $\mu 2(n)$, of a packet in one slot with the set $\left\{b^{(OUT2)}, \Psi^{(OUT2)}\right\}$, where $\Psi^{(OUT2)}$ is the set of all the possible numbers of packet services in one slot, while

$\underline{b}^{(OUT2)}$ is the service probability array, whose generic element is defined as:

$$b_{[\delta]}^{(OUT2)} = \Pr\{\mu 2(n) = \delta\} \tag{11}$$

Now, let us move to model the considered SDN/NFV node. In order to avoid defining a too-complex Markov chain modeling the time behavior of all the queues simultaneously, we leverage the peculiarity that all the inside-function queues are identically distributed, and the same occurs for the NIC queues. Therefore, once focused on an inside-function queue Q1 and the relative NIC queue Q2, we indicate the aggregate of the other queues loading the same NIC queue Q2 as $\overline{Q1}$, while the aggregate of the other NICs will be indicated as $\overline{Q2}$. Since the behaviors of the processes $S^{(Q1)}(n)$ and $S^{(Q2)}(n)$ also depend on the states of the "external inside-function queue aggregates" $\overline{Q1}$ and $\overline{Q2}$, we will derive the system model iteratively. More specifically, at each iteration, we will calculate the steady-state probability arrays for the two processes $S^{(Q1)}(n)$ and $S^{(Q2)}(n)$, let us indicate them as $\underline{\pi}^{(Q1)}$ and $\underline{\pi}^{(Q2)}$. These arrays will be taken as input of the next iteration to directly represent the probability arrays of the external queues $\overline{Q1}$ and $\overline{Q2}$, and the iterative process will be concluded when their Euclidean norms converge. Therefore, in order to model the whole system shown in Fig. 2, we use a Markov chain defined as $S^{(\Sigma)}(n) = \left(S^{(IN1)}(n), S^{(Q1)}(n), S^{(Q2)}(n)\right)$, where $S^{(IN1)}(n)$ is the state of the underlying Markov chain of the arrival process $\Lambda 1(n)$; $S^{(Q1)}(n)$ is the number of packets that are present in the queue $Q1$ at the generic slot n; $S^{(Q2)}(n)$ is the number of packets that are present in the queue $Q2$ at the generic slot n.

In order to define the transition probability matrix of the considered system, we assume the following sequence of events within the generic slot:

- at the beginning of the slot, the Processor Arbiter assigns the processor portions to the inside-function queues according to the NARR strategy;
- then the underlying Markov chain of the arrival process $\Lambda 1(n)$ changes its value, and new packets enter the queue Q1;
- then a number of packets leave the queues Q1 and $\overline{Q1}$, and enter the queue Q2;
- then a number of packets leave the queue Q2;
- finally, the system state is observed, and the process $S^{(\Sigma)}(n) = \left(S^{(IN1)}(n), S^{(Q1)}, S^{(Q2)}(n)\right)$ is updated.

Let us consider two generic states, $s'_\Sigma = \left(s'_{IN1}, s'_{Q1}, s'_{Q2}\right)$ and $s''_\Sigma = \left(s''_{IN1}, s''_{Q1}, s''_{Q2}\right)$, representing the start and the arrival states of a generic system transition from the slot n to the slot $n+1$. We define the generic element of the transition probability matrix as follows:

$$P^{(\Sigma)}_{[s'_\Sigma, s''_\Sigma]} = P^{(IN1)}_{[s'_{IN1}, s''_{IN1}]} \cdot P^{(QT)}_{[(s'_{Q1}, s'_{Q2}),(s''_{Q1}, s''_{Q2})]}\left(s''_{IN1}\right) \tag{12}$$

where:

- $P^{(IN1)}$ is the transition probability matrix of the underlying Markov chain of $\Lambda^{(IN1)}_{m,l}(n)$;
- $P^{(QT)}\left(s''_{IN1}\right)$ is the transition probability matrix of the 2-dimensional process $S^{(QT)}(n) \equiv \left(S^{(Q1)}(n), S^{(Q2)}(n)\right)$; According to the event sequence listed above, its evolution from the slot n to the slot $n+1$ depends on the new state of the arrival process. Its definition is:

$$P^{(QT)}_{[(s'_{Q1}, s'_{Q2}),(s''_{Q1}, s''_{Q2})]}\left(s''_{IN1}\right) = \\ = \Pr\left\{ \begin{matrix} S^{(Q1)}(n+1) = s''_{Q1}, \\ S^{(Q2)}(n+1) = s''_{Q2} \end{matrix} \middle| \begin{matrix} S^{(Q1)}(n) = s'_{Q1}, \\ S^{(Q2)}(n) = s'_{Q2} \\ S^{(IN1)}(n+1) = s''_{IN1} \end{matrix} \right\} \tag{13}$$

Now, let us apply the total probability theorem by considering all the possible arrivals in the queue Q1, i.e. for each $\alpha_1 \in \Psi^{(IN1)}$; all the possible states of the external queue $\overline{Q1}$, i.e. for each $q'_1 \in \{0,\ldots,(M-1)\cdot K_1\}$; all the possible numbers of arrivals to the Q2 queue from the Q1 queue, i.e. for each $\delta_1 \in \{0,\ldots,s'_{Q1}\}$; all the possible numbers of arrivals to the Q2 queue from the $\overline{Q1}$ queue, i.e. for each $\overline{\delta_1} \in \{0,\ldots,\overline{q'_1}\}$:

$$P^{(QT)}_{[(s'_{Q1}, s'_{Q2}),(s''_{Q1}, s''_{Q2})]}\left(s''_{IN1}\right) = \\ = \sum_{\alpha_1 \in \Psi^{(IN1)}} \Pr\left(\Lambda 1(n) = \alpha_1 \mid s''_{IN1}\right) \cdot \sum_{\overline{q'_1}=0}^{(M-1)\cdot K_1} \pi^{(\overline{Q1})}_{[\overline{q'_1}]} \cdot \\ \cdot \sum_{\delta_1=0}^{s'_{Q1}} \left[\Pr\left(\delta_1 \middle| s'_{Q1}, s'_{Q2}, \overline{q'_1}\right) \cdot I\left(s'_{Q1}, s''_{Q1}, \alpha_1, \delta_1, K_1\right)\right] \cdot \\ \cdot \sum_{\overline{\delta_1}=0}^{\overline{q'_1}} \Pr\left(\overline{\delta_1} \middle| s'_{Q1}, s'_{Q2}, \overline{q'_1}\right) \cdot \sum_{\delta_2 \in \Psi^{(OUT2)}} \left[b^{(OUT2)}_{[\delta_2]} \cdot I\left(s'_{Q2}, s''_{Q2}, \alpha_2, \delta_2, K_2\right)\right] \tag{14}$$

where:

- $\Pr\left(\Lambda 1(n) = \alpha_1 \mid s''_{IN1}\right)$ is the arrival probability to the Q1 queue. By definition of the SBBP arrival process $\Lambda 1(n)$, we have:

$$\Pr\left(\Lambda 1(n) = \alpha_1 \mid s''_{IN1}\right) = B^{(IN1)}_{[s''_{IN1}, \alpha_1]} \tag{15}$$

- $I\left(s'_Q, s''_Q, \alpha, \delta, K\right)$ is a Boolean indicator function on the possibility for a queue state to move from the value s'_Q to the value s''_Q, when α arrivals and δ departures occur, that is:

$$I\left(s'_Q, s''_Q, \alpha, \delta, K\right) = \\ = \begin{cases} 1 & \text{if } s''_Q = \max\{\min\{s'_Q + \alpha, K\} - \delta, 0\} \\ 0 & \text{otherwise} \end{cases} \tag{16}$$

K being the queue system size.

- α_2 is the total number of arrivals to the NIC queue Q2 in the considered slot, that is, $\alpha_2 = \delta_1 + \overline{\delta_1}$;
- $\underline{\pi}^{(\overline{Q1})}$ is the array containing the steady-state probabilities of the external inside-function queue aggregate $\overline{Q1}$. It can be calculated as the convolution among the steady-state probability arrays of the $M-1$ inside-function queues other than Q1. Since all the inside-function queues are identically distributed, we have:

$$\underline{\pi}^{(\overline{Q1})} = \bigotimes_{m=1}^{M-1} \underline{\pi}^{(Q1)} \tag{17}$$

- $P\left(\delta_1 \middle| s'_{Q1}, s'_{Q2}, \overline{q'_1}\right)$ is the probability that the processor portion assigned to the $Q1$ queue is δ_1. According to the definition of the NARR strategy described in the previous section, it depends on the state of the other NICs and, in particular, on the state of the NIC with the highest queue, because this determines the q_{ref} value. For this reason, we define $\underline{v2}$, a $(L-1)$-dimensional array containing the state of the external NIC queues. Now, by applying the theorem of total probability on all the possible values of $\underline{v2}$ in their state space $\Im^{(v2)}$, we have:

$$P\left(\delta_1 \middle| s'_{Q1}, s'_{Q2}, \overline{q'_1}\right) = \sum_{\underline{v2} \in \Im^{(V2)}} \Pr\left\{S^{(\overline{Q2})}(n) = \underline{v2}\right\} \cdot \Pr(\delta_1 \middle| s'_{Q1}, s'_{Q2}, \overline{q'_1}, \underline{v2}) \tag{18}$$

The first probability in (18) can be calculated by considering that all the NIC queues are identically distributed, and therefore we have:

$$\Pr\left\{S^{(\overline{Q2})}(n) = \underline{v2}\right\} = \prod_{l=1}^{L-1} \pi^{(Q2)}_{[v2_{[l]}]} \tag{19}$$

The second probability in (18) will be derived in the Appendix for the sake of readability.

- $P\left(\overline{\delta_1} \middle| s'_{Q1}, s'_{Q2}, \overline{q'_1}\right)$ is the probability that the processor portion assigned to the aggregate of the external inside-function queues, $\overline{Q1}$, is $\overline{\delta_1}$. It can be calculated like the probability in (18), that is:

$$P\left(\overline{\delta_1} \middle| s'_{Q1}, s'_{Q2}, \overline{q'_1}\right) = \sum_{\underline{v2} \in \Im^{(V2)}} \Pr\left\{S^{(\overline{Q2})}(n) = \underline{v2}\right\} \cdot \Pr(\overline{\delta_1} \middle| s'_{Q1}, s'_{Q2}, \overline{q'_1}, \underline{v2}) \tag{20}$$

where $\Pr\left\{S^{(\overline{Q2})}(n) = \underline{v2}\right\}$ has already been calculated in (19), while $\Pr(\overline{\delta_1} \middle| s'_{Q1}, s'_{Q2}, \overline{q'_1}, \underline{v2})$ will be derived in the Appendix.

From the transition probability matrix in (12) we can calculate the steady-state probability array of the whole system, $\underline{\pi}^{(\Sigma)}$. Finally, we can calculate the marginal steady-state probability arrays of the two queues Q1 and Q2:

$$\pi^{(Q1)}_{[s_{Q1}]} = \Pr\left\{S^{(Q1)}(n) = s_{Q1}\right\} = \sum_{s_{IN1} \in \Im^{(IN1)}} \sum_{s_{Q2}=0}^{K_2} \pi^{(\Sigma)}_{[s_{IN1}, s_{Q1}, s_{Q2}]} \tag{21}$$

$$\pi^{(Q2)}_{[s_{Q2}]} = \Pr\left\{S^{(Q2)}(n) = s_{Q2}\right\} = \sum_{s_{IN1} \in \Im^{(IN1)}} \sum_{s_{Q1}=0}^{K_1} \pi^{(\Sigma)}_{[s_{IN1}, s_{Q1}, s_{Q2}]} \tag{22}$$

that will be used in the next section to derive some numerical results.

IV. Numerical results

Let us now apply the analytical model introduced in the previous section to evaluate performance of an SDN/NFV node that uses the NARR processor-sharing strategy, with the target of evaluating the performance gain achieved as compared with a *round robin* (*RR*) processor-sharing strategy taken as reference. Since in the RR case the node has only M processor queues, one for each VNF, we set the maximum size of each VNF queue equal to $K^{(F)} = L \cdot K^{(F)}_{Ins}$, where $K^{(F)}_{Ins}$ represents the size of each inside-function queue, already defined so far for the NARR strategy. The RR strategy applies the classical round-robin scheduling policy to serve the M function queues, i.e. it serves each function queue with a rate $\mu^{(F)}_{RR} = \mu^{(P)}/M$.

In this numerical analysis, we consider a node with $M = 4$ VNFs, and $L = 3$ output NICs, and loaded by $N_F = 12$ flows, each characterized by a different 2-uple {*requested VNF*, *output NIC*}. Each flow is represented by an ON-OFF model. When a flow is in the OFF state, no packets arrive to the node from it; instead, when it is in ON, packets arrive with an average rate λ_{ON}. Each packet is assumed with an exponential distributed size with a mean of $\overline{P} = 1$ kbyte.

Let us stress that the model is general and it can accept any kind of real input traffic trace.

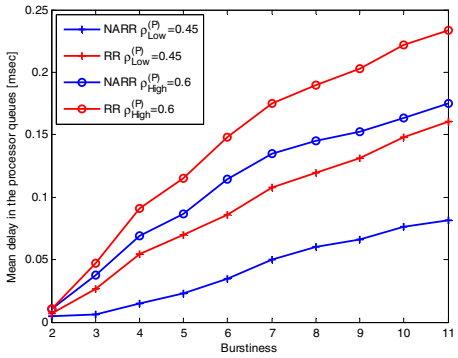
In the whole analysis we will consider the same ON-OFF cycle duration, $\tau = \overline{T}_{OFF} + \overline{T}_{ON} = 5\ m\text{sec}$, while we will vary the burstiness in the range [2, 11]. Burstiness of ON-OFF sources is defined as $b = \lambda_{ON}/\lambda_{Mean}$, where λ_{Mean} is the mean emission rate, in the following assumed equal to λ_{Mean} = 117 Mbit/s. Moreover, we have assumed a mean duration of the ON periods equal to $\overline{T}_{ON}$ = 25 slots. Now, indicating the probability of the ON state as π_{ON}, and taking into account that $\lambda_{Mean} = \lambda_{ON} \cdot \pi_{ON}$ and $\pi_{ON} = \overline{T}_{ON}/\left(\overline{T}_{OFF} + \overline{T}_{ON}\right)$, we have:

$$b = \left(\overline{T}_{OFF} + \overline{T}_{ON}\right)/\overline{T}_{ON} \tag{23}$$

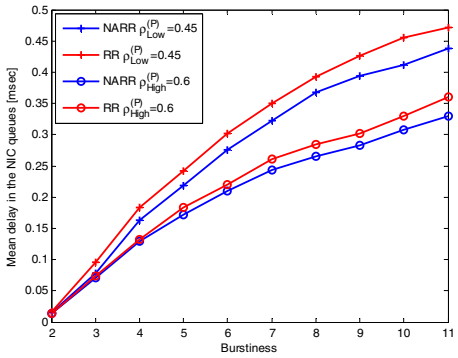
Therefore, the transition probability matrix of the SBBP model of one input traffic flow is:

$$P^{(IN1)} = \begin{bmatrix} \dfrac{(b-1)\overline{T}_{ON} - 1}{(b-1)\overline{T}_{ON}} & \dfrac{1}{(b-1)\overline{T}_{ON}} \\ \dfrac{1}{\overline{T}_{ON}} & \dfrac{\overline{T}_{ON} - 1}{\overline{T}_{ON}} \end{bmatrix} \tag{24}$$

Moreover, we assumed the following arrival probability matrix:



(a) Mean delay in the processor queues



(b) Mean delay in the NIC queues

Figure 3: Queue delay comparison

$$B^{(IN1)} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.28 & 0.44 & 0.28 \end{bmatrix} \quad (25)$$

Consequently, the mean arrival rate during the ON periods is of $\lambda_{ON} = 3$ packets/slot.

As far as the NICs are concerned, we have considered an output rate of $\mu^{(NIC)} = 937$ Mbit/s, so having an utilization coefficient on each NIC of $\rho^{(NIC)} = 0.5$, and a queue size of $K^{(NIC)} = 250$ packets. The service process has been defined on a set of possible numbers of packet services per slot given by $\Psi^{(OUT2)} = \{0,1,\ldots,\mu_{MAX}^{(NIC)}\}$, with $\mu_{MAX}^{(NIC)} = \lceil 2 \cdot (\Delta \cdot \mu^{(NIC)} / \overline{P}) \rceil$. Given the exponential distribution of the packet size, the per-slot service probability array has been calculated assuming a Poisson distribution.

Regarding the processor, we considered a size of each inside-function queue of $K_{Ins}^{(F)} = 80$ packets. Finally, we have considered that the NARR strategy uses five ranges on both the inside-function queues and the NIC queues.

We have analyzed two different processor cases. In the first case, we considered a processor that is able to process $\mu^{(P)} = 293$ kpackets/s, while in the second case we assumed a processor rate of $\mu^{(P)} = 391$ kpackets/s. Therefore, defining the processor utilization coefficient as follows:

$$\rho^{(P)} = (N_F \cdot \lambda_{Mean}) / \mu^{(P)} \quad (26)$$

$N_F \cdot \lambda_{Mean}$ being the total mean arrival rate to the node, the two considered cases are characterized by a processor utilization coefficient of $\rho_{Low}^{(P)} = 0.45$ and $\rho_{High}^{(P)} = 0.6$, respectively.

The achieved results are shown in Fig. 3, which presents the mean delay in both the processor queues and the NIC queues against the input traffic burstiness.

To derive the mean delay, let us first calculate the mean number of jobs present in an internal queue and in a NIC queue.

$$E\{Q_1\} = \sum_{s_\Sigma \in \Im^{(\Sigma)}} s_{Q1} \cdot \pi_{[s_\Sigma]}^{(\Sigma)} \quad (27)$$

$$E\{Q_2\} = \sum_{s_\Sigma \in \Im^{(\Sigma)}} s_{Q2} \cdot \pi_{[s_\Sigma]}^{(\Sigma)} \quad (28)$$

where $s_\Sigma = [s_{IN1}, s_{Q1}, s_{Q2}]$ and $\Im^{(\Sigma)}$ is the set of all the possible states of the system.

According to the Little theorem, the mean time spent, respectively, in an internal queue and in a NIC queue can be derived as:

$$E\{T_1\} = \frac{E\{Q_1\}}{\overline{J_1}} \quad (29)$$

$$E\{T_2\} = \frac{E\{Q_2\}}{\overline{J_2}} \quad (30)$$

where $\overline{J_1}$ and $\overline{J_2}$ are the correspondent mean job arrival rates.

For the internal queue, $\overline{J_1} = \lambda_{Mean} \cdot (1 - p_{loss1})$, where:

$$p_{loss1} = \sum_{s_\Sigma \in \Im^{(\Sigma)}} \pi_{[s_\Sigma]} \cdot \sum_{\alpha_1 \in \Psi^{(IN1)}} B_{[s_{IN1},\alpha_1]}^{(IN1)} \cdot I_{loss}(s_{Q1}, \alpha_1, K_1) \quad (31)$$

and:

$$I_{loss}(s_{Q1}, \alpha_1, K_1) = \begin{cases} 1 & \text{if } s_{Q1} + \alpha_1 > K_1 \\ 0 & \text{otherwise} \end{cases} \quad (32)$$

while for the NIC:

$$\overline{J_2} = \sum_{s'_\Sigma} \pi_{[s'_\Sigma]}^{(\Sigma)} \cdot \sum_{s''_\Sigma} P_{[s'_\Sigma, s''_\Sigma]}^{(\Sigma)} \cdot \sum_{\delta_2 \in \Psi^{(OUT2)}} b_{[\delta_2]}^{(OUT2)} \cdot (s''_{Q2} + \delta_2 - s'_{Q2}) \quad (33)$$

As expected, the mean length of the processor queues and the NIC queues increase with the burstiness. Moreover, the cases with the highest utilization coefficient in the processor queues present higher delays in those queues. On the contrary, the mean delay in the NIC queues is higher in the case of $\rho_{Low}^{(P)} = 0.45$. In addition we can notice that the gain achieved with the NARR strategy is evident in both the queues, and for $\rho_{Low}^{(P)} = 0.45$ and $\rho_{High}^{(P)} = 0.6$, respectively, it reaches the values of 2.49% and 8.29% in the NIC queues, and even 49.26% and 25.21% in the processor queues.

Conclusions and Future Work

This paper proposes an analytical model of an SDN/NFV node, and is, at the best of our knowledge, the first paper where

this is done. The considered node uses the NARR strategy, a processor sharing strategy proposed by the same authors in a previous work. Thanks to the proposed model, the strategy is evaluated through numerical results.

As a future work, the authors are working on extending the proposed model to capture the behavior of a whole network constituted by the interconnection of different SDN/NFV nodes. In addition, another evolution of the model is to capture also network services, realized by chaining different VNFs by predefined graphs.

ACKNOWLEDGEMENTS

This work has been partially supported by the INPUT (In-Network Programmability for next-generation personal cloUd service supporT) project funded by the European Commission under the Horizon 2020 Programme (Call H2020-ICT-2014-1, Grant no. 644672).

APPENDIX

In this Appendix we derive the two probability terms $P\left(\delta_1 \middle| s'_{Q1}, s'_{Q2}, \overline{q'_1}\right)$ and $P\left(\overline{\delta_1} \middle| s'_{Q1}, s'_{Q2}, \overline{q'_1}\right)$.

In order to calculate the first probability term, we indicate the processor portion that is assigned to all the flows that will leave the node through the considered NIC, expressed in packets/s, as $\rho_T(s'_{Q2}, \underline{v2})$, where we have explicitly indicated its dependence on the queue state of the considered NIC, s'_{Q2}, and of the other NICS, $\underline{v2}$. As defined in (8), it can be derived as follows:

$$\rho_T(s'_{Q2}, \underline{v2}) = \\ = \frac{\varphi_{ref}(\underline{v2}, s'_{Q2}) - \varphi_2\left(s'_{Q2}\right)}{L \cdot \varphi_{ref}(\underline{v2}, s'_{Q2}) - \left[\varphi_2\left(s'_{Q2}\right) + \varphi_2\left(\hat{s}_{\overline{Q2}}(\underline{v2})\right)\right]} \cdot \mu^{(P)} \quad (34)$$

Therefore, the maximum number of packets that can be served by the processor for all the flows that will leave the node through the considered NIC is:

$$\delta_T(s'_{Q2}, \underline{v2}) = \min\left\{\rho_T(s'_{Q2}, \underline{v2}), K_1\right\} \quad (35)$$

Moreover, from (35) we can derive the processor sub-portion that is assigned to the flows that receive the considered VNF F_m and leave the node through the NIC *l*:

$$\delta_*(s'_{Q1}, s'_{Q2}, \overline{q'_1}, \underline{v2}) = \\ = \min\left\{\frac{\varphi_1\left(s'_{Q1}\right)}{\varphi_1\left(s'_{Q1}\right) + \varphi_1\left(\overline{q'_1}\right)} \cdot \delta_T(s'_{Q2}, \underline{v2}), K_1\right\} \quad (36)$$

where $\varphi_1\left(s'_{Q1}\right)$ has been defined as in (5), but with a range size of Δ_1.

Since the value of $\delta_T(s'_{Q2}, \underline{v2})$ derived as in (35) can result not an integer, it will be rounded to the nearest integer values with a probability equal to their distances from the real value, we can derive the required probability as follows:

$$P(\delta_1 | s'_{Q1}, s'_{Q2}, \overline{q'_1}, \underline{v2}) = \\ = \begin{cases} 1 - \left|\delta_*(s'_{Q1}, s'_{Q2}, \overline{q'_1}, \underline{v2}) - \delta_1\right| & \text{if } \left|\delta_*(s'_{Q1}, s'_{Q2}, \overline{q'_1}, \underline{v2}) - \delta_1\right| < 1 \\ & \text{and } \left(\delta_1 \neq 0 \text{ or } \overline{q'_1} \neq 0\right) \\ 1 & \text{if } \delta_1 = 0 \text{ and } \overline{q'_1} = 0 \\ 0 & \textit{otherwise} \end{cases} \quad (37)$$

Likewise, the term $P\left(\overline{\delta_1} \middle| s'_{Q1}, s'_{Q2}, \overline{q'_1}\right)$ can be calculated as follows:

$$P(\overline{\delta_1} | s'_{Q1}, s'_{Q2}, \overline{q'_1}, \underline{v2}) = \\ = \begin{cases} 1 - \left|\overline{\delta_*}(s'_{Q1}, s'_{Q2}, \overline{q'_1}, \underline{v2}) - \overline{\delta_1}\right| & \text{if } \left|\overline{\delta_*}(s'_{Q1}, s'_{Q2}, \overline{q'_1}, \underline{v2}) - \overline{\delta_1}\right| < 1 \\ & \text{and } \left(\overline{\delta_1} \neq 0 \text{ or } \overline{s'_{Q1}} \neq 0\right) \\ 1 & \text{if } \overline{\delta_1} = 0 \text{ and } \overline{s'_{Q1}} = 0 \\ 0 & \textit{otherwise} \end{cases} \quad (38)$$

where:

$$\overline{\delta_*}(s'_{Q1}, s'_{Q2}, \overline{q'_1}, \underline{v2}) = \\ = \min\left\{\frac{\varphi_1\left(\overline{q'_1}\right)}{\varphi_1\left(s'_{Q1}\right) + \varphi_1\left(\overline{q'_1}\right)} \cdot \delta_T(s'_{Q2}, \underline{v2}), K_1 \cdot (M-1)\right\} \quad (39)$$

REFERENCES

[1] A. Manzalini et al., "Software-Defined Networks for Future Networks and Services," White Paper based on the IEEE Workshop SDN4FNS, available at *http://sites.ieee.org/sdn4fns/ whitepaper/*, 2014.

[2] White paper on "Software-Defined Networking: The New Norm for Networks", available at *https://www.opennetworking.org/*, 2012.

[3] White paper on "Network Functions Virtualisation", available at *http://portal.etsi.org/NFV/NFV_White_Paper.pdf*, 2012.

[4] A. Lombardo, *et al.*, "An Open Framework to Enable NetFATE (Network Functions At The Edge)," in Proc. of Mission 2015, London, UK, April 13-17, 2015.

[5] G. Faraci and G. Schembra, "An analytical model to design and manage a green SDN/NFV CPE node," IEEE Transactions on Network and Service Management, vol. 12, no. 3, pp. 435–450, Sept 2015.

[6] H. Moens, F. De Turck, "VNF-P: A model for efficient placement of virtualized network functions," in Proc. of CNSM 2014, Rio de Janeiro, Brazil, November 17-21, 2014.

[7] G. Faraci, A. Lombardo, G. Schembra, "A Processor-Sharing Scheduling Strategy for NFV Nodes," accepted for publication on Hindawi Journal of Electrical and Computer Engineering, Special Issue on Design of High Throughput and Cost-Efficient Data Center Networks, Volume 2016, Article ID 3583962, http://dx.doi.org/10.1155/2016/3583962.

[8] Kleinrock, L. (1967). "Time-shared Systems: A theoretical treatment" (PDF). Journal of the ACM, vol. 14, no. 2.

[9] ETSI NFV GS, Network Function Virtualization (NFV) Management and Orchestration, NFV-MAN 001 v0.8.1, Nov 2014.

[10] ETSI NVF GS, Network Functions Virtualization (NFV) Infrastructure Overview, NFV-INF 001 V1.1.1, Jan 2015.