




Article

Modeling a Linux Packet-Capturing System with a Queueing System with Vacations

Luis Zabala ¹, Josu Doncel ^{2,*} and Armando Ferro ¹

¹ Department of Communications Engineering, University of the Basque Country, UPV/EHU, 48013 Bilbao, Spain

² Department of Mathematics, University of the Basque Country, UPV/EHU, 48940 Leioa, Spain

* Correspondence: josu.doncel@ehu.eus

Abstract: Monitoring the evolution of the state of networks is an important issue to ensure that many applications provide the required quality of service. The first step in network-monitoring systems consists of capturing packets; that is, packets arrive at the system through a network interface card and are placed into system memory. Then, in this first stage, and usually in relation to the operating system, packets are treated and transferred from the capturing buffer to a higher-layer processing, for instance, to be analyzed in the next step of the system. In this work, we focus on the capturing stage. In particular, we focus on a Linux packet-capturing system. We model it as a single server queue. Taking into account that the server can be in charge not only of the capturing process but also of other tasks, we consider that the queue has vacations, i.e., there is some time when the capturing process cannot be carried out. We also assume that the queue has a finite buffer. We consider three different models and present a rigorous analysis of the derived Markov chain of each of the models. We provide standard performance metrics in all cases. We also evaluate the performance of these models in a real packet-capture probe.

Keywords: single server queue with vacations; Linux packet capturing; network monitoring system

MSC: 60K30; 68M20; 68J28



Citation: Zabala, L.; Doncel, J.;

Ferro, A. Modeling a Linux

Packet-Capturing System with a

Queueing System with Vacations.

Mathematics **2023**, *11*, 1663. [https://](https://doi.org/10.3390/math11071663)

doi.org/10.3390/math11071663

Academic Editor: János Sztrik

Received: 28 February 2023

Revised: 27 March 2023

Accepted: 28 March 2023

Published: 30 March 2023



Copyright: © 2023 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article

distributed under the terms and

conditions of the Creative Commons

Attribution (CC BY) license ([https://](https://creativecommons.org/licenses/by/4.0/)

[creativecommons.org/licenses/by/](https://creativecommons.org/licenses/by/4.0/)

[4.0/](https://creativecommons.org/licenses/by/4.0/)).

1. Introduction

There is a wide range of applications where the interest resides in identifying the following facts: (i) when the structure of a network changes over time, or (ii) whether some subnetwork is different from the network in which it is contained in some meaningful way. Network monitoring is known to be the field that identifies such changes [1]. Telecommunication companies currently have a big interest in understanding the performance of modern network-monitoring systems [2].

In a network-monitoring system, the first phase consists of capturing the packets from the network to obtain the raw measurement data. This data is then analyzed and presented to the network operators. Recently, several solutions have been presented in the literature to achieve good performance in network-monitoring systems; for instance PF_RING [3], PacketShader [4], Netmap [5], PFQ [6], DPDK [7], OpenOnLoad [8], and HPCAP [9].

However, our goal is to develop a mathematical model to study the performance of network-monitoring systems. In this work, we focus on the capturing phase of a network-monitoring system. The object of our research is to predict the performance of that packet-capturing system by using the solution of a queueing model. Our modeling proposal is a queueing system formed by a single queue with one server. We are interested in the case of network devices with limited resources. Therefore, we will assume that the system only has one processor and one network card. Furthermore, we consider that the network-monitoring system under study needs to perform other tasks when the capturing

phase is activated. This means that the server that is in charge of the capturing phase needs to stop this work to deal with other tasks. This is the case, for instance, when the server needs to capture and analyze the data [10]. Here arises the concept of vacation, which represents the period of temporary absence of the server. With all this, we will estimate some performance metrics, such as throughput and probability of packet loss, under different incoming packet traffic conditions and vacation scenarios.

There is a considerable amount of work that models telecommunications networks by using queueing theory, ranging from the very old to the very new systems [11]. Among those published in recent years, we can mention [12], in which an analytical model was elaborated upon to predict some traffic characteristics, quality of service (QoS) indicators, and the overall network performance. In [13] it is proposed a conceptual model of overall telecommunication system with a queueing system; this analytical model was used to solve aspects such as dimensioning of the network and predicting the QoS.

Other even more recent works are related to fifth-generation (5G) networks. In [14] it is presented a queueing model to control the number of user plane functions (UPF) in a 5G network; the system is described by a two-dimensional continuous time Markov chain (CTMC). The authors in [15] developed an analytical queueing model to analyze the network slicing strategies for 5G networks; the solutions and the system's performance metrics were derived from a two dimensional Markov chain. The authors in [16] formulated a queueing-based model and used it at the network orchestrator to optimally perform a virtual network function (VNF) placement and a resource allocation for the support of the vertical's requirements in 5G networks. The authors in [17] studies how to determine various performance metrics (system's waiting time, the throughput and number of resources required to meet users' needs) of a vehicular ad hoc network (VANET) with two queueing theory models, namely the M/M/c and the M(N)/M/c models. We can also cite some work on software-defined networking (SDN) modeling. [18] assessed the quality of SDN operation by estimating the average values of packet delay time in M/G/1 and G/G/1 systems for SDN. The authors in [19] proposed an analytical Markovian queueing model based on M/M/c to manage the traffic that transit via the 5G access networks to SDN architecture.

However, we notice the literature regarding mathematical modeling in the field of packet-capturing systems is scarce. For instance, [20] presents the Linux packet-receiving system as a combination of (i) the token bucket algorithm that fits to the network card and device driver-receiving process, and (ii) a queueing system that models the rest of the packet-receiving process. In [21,22], firewalls are represented by a finite queueing system with multiple stages, and the first stage refers to the packet-capturing phase. Finally, another interesting approach is [23] which proposes an analytical model to formulate the performance of the main techniques adopted by three high-speed packet I/O frameworks, such as Netmap, Intel DPDK, and PF_RING ZC.

The main difference of our model with related works is that, in our work, we model a Linux packet-capturing system by using a queueing system with vacations. Queueing systems with vacations have been studied in queueing theory [24,25], but to the best of our knowledge, this work is the first to consider vacations in a model for the capturing phase of a network-monitoring system.

We present three different models. In all cases, there is a limited waiting room in the queue. However, we now present the differences between the models we study. For the sake of clarity of the presentation, we start with the simplest model and we finish with the most difficult one.

- In the first model, M1, we consider exponential service times and the fact that there is not a limitation on the number of packets that can be served. This means that if the capturing process is active, it does not end until there is no packet left in the buffer. It is at this time when the system returns to vacation.

- In the second model, M2, we consider exponential service times with a limited budget of the capturing phase. The budget indicates the maximum number of packets that can be served without a vacation.
- In the last model, M3, we consider arbitrary service times and the fact that the budget of the capturing phase is limited.

The main contribution of this work is to develop the Markov chain related to each of the queueing models we consider and to determine analytical expressions of the performance metrics of all the systems. It is important to remark that model M1 generalizes the work of [26] and model M2 generalizes the work of [27]. In both M1 and M2, we consider an initial phase, which we call the hardware interrupt, which is not considered in [26,27].

The rest of the article is organized as follows. In Section 2, the mechanism of the Linux packet-capturing system and the proposed queueing model that represents it are described. Then, Sections 3–5 present the model analysis for three cases of interest: (1) analysis for exponential time distributions and infinite budget, (2) analysis for exponential time distributions and finite budget, and (3) analysis with time distributions of a general type and finite buffer, respectively. The model validation and the obtained results are discussed in Section 6. Finally, we conclude the paper in Section 7.

2. Model Description

This section presents the general aspects of the model that will represent the Linux packet-capturing system. Our model is based on a finite queueing system with vacations. However, before detailing the characteristics of the model, it is necessary to explain how that packet-receiving process works. We will focus on the mechanism called New API (NAPI).

2.1. Linux Network Subsystem

Figure 1 shows the trip of a packet from its arrival on a network interface to its delivery to the user-space application. First, packets arrive at the network interface card (NIC) from the network. Then, packets are transferred into a ring buffer in kernel memory via direct memory access (DMA). It should be pointed out that the DMA transfer does not require any CPU consumption. If the network card drives many packets into the ring buffer but the kernel does not extract them, the buffer will eventually fill up, and new, incoming packets can be rejected.

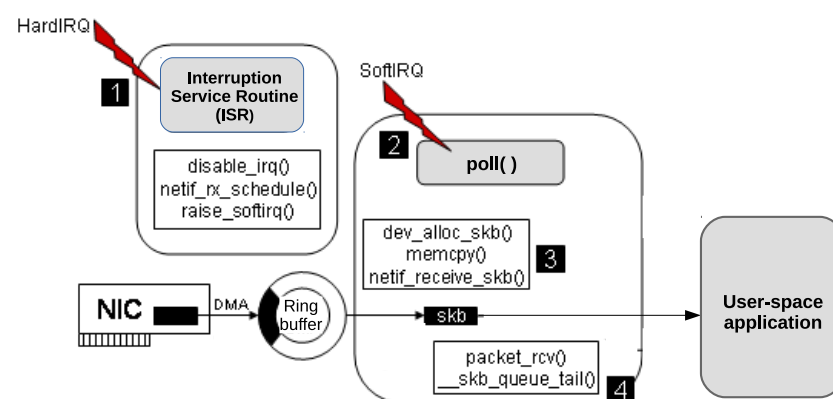


Figure 1. Linux packet-capture mechanism.

Traditionally, when a packet was written to kernel memory via DMA, an interrupt request (IRQ) was generated to indicate, to the rest of the system, that data is ready to be processed. However, if large numbers of packets arrive, this can lead to a large number of IRQs. The more IRQs are generated, the less CPU time is available for higher-level tasks (such as user processes). In order to avoid the risk to consume almost all of the CPU time by a high number of IRQs, the Linux network subsystem addresses this issue through NAPI. Drivers based on NAPI use a mechanism that mixes polling and interrupt-

driven processing [28]. Therefore, there are two parts: the hardware interrupt (hardirq, henceforth) and the software interrupt (softirq, henceforth). The hardirq is involved by the IRQ handling. It is the fast part (or top half), and it only performs critical actions that cannot be delayed. On the contrary, the softirq is the slower part, or bottom half, it polls the ring buffer to process network packets, and it avoids looping for too much time in a hardirq handler.

When there are no packets in the ring buffer and the first one is written via DMA, the device raises the hardirq that is assigned to it. The interruption service routine (ISR), which runs when a hardirq is raised, is short and performs the following operations.

- Hardirqs related to the network card are disabled.
- The network interface is registered in the poll list of the CPU that is executing the hardirq handler. There is only one poll list in each CPU.
- A function is called to schedule the execution of a softirq (netif_rx_schedule() in Figure 1). The softirq will be triggered later, when the Linux process scheduler allocates CPU resources to it.
- The correct reception of the interruption is also acknowledged to the network card.

When the softirq is invoked, ring buffer packets begin to be treated. The softirq poll() loop aims to transfer packets from the ring buffer to the upper-level structure that is ready to receive data. As Figure 1 shows, this action involves calling several functions (marked with 3 and 4 in Figure 1), as well as using structures named skb [29]. When a packet is dequeued from the ring buffer, its corresponding packet descriptor needs to be reinitialized and refilled.

Since one CPU can handle multiple network cards, during the softirq, the CPU polls each device that is registered in its poll list. In each poll, there is a maximum number of packets that the softirq can process. That limitation is called weight, and its typical value is 64. When the weight is reached, the softirq moves to poll the next device. If the poll does not consume its entire weight, this means that all the packets from that device have already been transferred and, therefore, that device should be removed from the poll list.

Furthermore, there exist other limitations for the softirq.

- The total number of packets that the softirq can handle is limited by the parameter named "budget". Its default value is 300. When this maximum is reached, the softirq finishes.
- The softirq will still be bounded by a time limit of 2 jiffies, regardless of the assigned budget [30].

If the softirq finishes and there are still packets to pull out, a new softirq is scheduled. On the contrary, once there is no more work to do (that is, there are no more devices registered in the poll list), the NAPI subsystem is disabled and IRQs from the device are reenabled; in other words, there is no scheduled softirq, and hardirqs are enabled again. The process will start back when the first packet arrives at the ring buffer.

2.2. Finite Queue with Vacations

The packet-capturing system described above is modeled by a single server queue with a finite capacity and vacations. We have the following assumptions for the model.

There is an arrival process with rate λ representing the incoming packets to the ring buffer (via DMA). In order to ensure that the analytical solution does not become unmanageable, we assume that packets arrive at the system by following a Poisson process with rate λ .

The ring buffer is represented by a finite waiting queue of size N (maximum number of packets in the buffer). In the real system, N depends on the device driver and the network card. If the buffer is full and new packets arrive, they will be rejected and therefore lost.

There is only one processor (or CPU) and one NIC to perform the capturing task. CPU consumptions are related to the service capacity of the queue, but we keep in mind that there are two types of processing: hardirq processing and softirq processing. For this reason, we define a hardirq service time (τ_H), i.e., the time required to run the ISR, with an associated rate μ_H , which indicates the mean number of hardirqs treated per second. We

also have a packet service time in softirq (τ_s), i.e., the time required to poll each packet, with an associated rate μ_s which indicates the mean number of packets treated per second in a softirq. Moreover, as we identify vacation times, i.e., time in which the CPU is idle or handling processes other than packet capture, we denote τ_v as vacation time and its associated rate as μ_v (mean number of vacations per second). As shown in Figure 2.

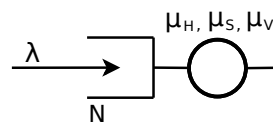


Figure 2. Finite queueing system to represent hardirq and softirq processing and vacations.

Let variable n be the number of packets in the steady state of the system ($0 \leq n \leq N$). Let variable m be the state of the server where $m = H$ indicates that the CPU is running a hardirq’s ISR, $m = V$ indicates that the CPU is on vacation, and a value of m between 1 and B indicates that the CPU is running a softirq and it is attending the m th packet in the current softirq. B is the value of the budget: the maximum number of packets that can be served in a softirq.

Figure 3 shows the overall state diagram of the system. The model aims to collect all its states and transitions in order to represent the behavior of the packet-capture system.

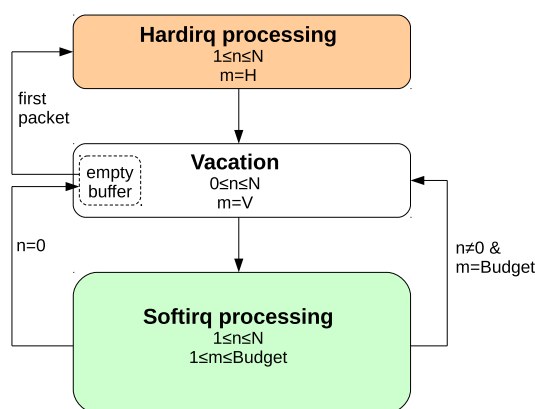


Figure 3. Diagram of processings and vacations.

Suppose we start from the empty buffer state ($n = 0, m = V$). As can be seen in Figure 3, the empty buffer state belongs to the set of vacation states, but we will consider it as special, because it is always the state prior to the start of a hardirq. The system remains there until the first packet (with rate λ) arrives—that is, when we move into the hardirq processing state, and specifically into the initial state of hardirq processing ($n = 1, m = H$). During the execution of the hardirq ISR, new packets can reach the system and n increases, but the system will remain within the set of states ($n, m = H$) where $1 \leq n \leq N$. These new packets will be accepted as long as the buffer is not completely filled. In case there is no more buffer space (state ($n = N, m = H$)), the new, incoming packets will be rejected, and the system will remain in the same state.

According to the Linux network processing based on NAPI, when the hardirq ends, the softirq is already scheduled, but its execution is not immediate. In this case, the softirq is not in the context of hardirq and it runs at a later time in ksoftirqd [29]. Therefore, in the state diagram of Figure 3, when the hardirq processing ends, we move to a vacation state ($n, m = V$) where $1 \leq n \leq N$ (notice that it is not possible to move from the hardirq processing to the empty buffer state). Again, the packet arrival remains active regardless of the state m . For this reason, during the vacation, n can increase to N at most.

When the vacation time expires, the transition between the final state of vacation ($n, m = V$) and the initial state of softirq ($n, m = 1$), where $1 \leq n \leq N$, takes place. The softirq processing is represented by a set of states (n, m) where $1 \leq n \leq N$ and

$1 \leq m \leq B$. During the softirq, new packets with rate λ can arrive and, therefore, transitions from (n, m) to $(n + 1, m)$ can occur, except in the blocking states (N, m) . The departure of processed packets is also possible, resulting in transitions from (n, m) to $(n - 1, m + 1)$ where $2 \leq n \leq N$ and $1 \leq m \leq B - 1$. Another important aspect to consider is the completion of the softirq for which there are three options.

- The first option involves the fact that the queue is emptied and, according to the Linux procedure, hardirq is enabled again. In this case, we move from the softirq processing to the empty buffer state (transition marked with $n = 0$ in Figure 3). This is a transition from the state $(1, m)$, where $1 \leq m \leq B$, to the state $(0, V)$.
- The second option is to reach the budget without having completely emptied the buffer, states (n, B) where $2 \leq n \leq N$. In this case, Linux schedules a new softirq without enabling hardirqs. We move from the softirq processing to a regular vacation state (transition marked with $n \neq 0$ and $m = B$ in Figure 3). This is a transition from the state (n, B) to a state $(n - 1, V)$ where $2 \leq n \leq N$.
- The time limit of 2 jiffies has elapsed. However, we will disregard this case for our model, since we have verified experimentally that this case occurs very rarely.

If we define our queueing model with Kendall’s notation, we can name it as a queue $M/G/1/N$ with limited service discipline, vacations, and a setup period. The queue $M/G/1/N$ with limited service discipline represents the softirq process, whereas the setup period corresponds to the hardirq process.

Taking all this into account, we propose the analysis of three particular cases of the general model presented here. We will start with the simplest case of the three, and end with the most complex. We will call them M1, M2, and M3 as follows:

- M1. Model with infinite budget ($B \rightarrow \infty$) and exponential distributions for hardirq- and softirq-processing times and vacations;
- M2. Model with finite budget and exponential distributions for hardirq- and softirq-processing times and vacations; and
- M3. Model with finite budget and general distributions for hardirq- and softirq-processing and vacation times.

3. Analysis for Exponential Time Distributions and Budget $B \rightarrow \infty$

The first model, named M1, is based on a finite queue model with a set of Markovian assumptions. First, we assume that network packets arrive at the capturing queue according to a Poisson process with rate λ . Then, the capture engine’s packet processing is represented by a single server which is devoted to that task in the active period and to other ones in the vacation period. We should remember that there is also a time to attend to the hardirq service routine. The size of the capturing buffer limits the number of packets to N at the queueing system. If there are N packets at the system, the new, incoming packets will be blocked, i.e., those packets do not enter the buffer because it is full.

Model M1 assumes that the packet service time (related to the softirq process) follows an exponential distribution with a mean value of $1/\mu_S$. The packet service time can be considered the time the system needs to process a packet and carry it to the user application. That time includes the network stack processing, which is mainly performed by the kernel. In model M1, the softirq process does not end until the queue is emptied. Therefore, we consider an exhaustive service discipline for the softirq.

As far as the hardirq and vacations are concerned, the hardirq service time is also characterized with an exponential distribution of mean value $1/\mu_H$ and, similarly, the vacation time is exponential with mean value $1/\mu_V$. The procedure followed by the hardirq and vacations is the same as the generic case explained in Section 2.2.

3.1. Markov Chain Related to the Capturing Stage

The first finite queueing model with vacations, M1, is based on a Markov chain with a state space $\mathcal{S} := \mathcal{N} \times \mathcal{M}$. The state of the system is $s = (n, m)$ with $n \in \mathcal{N} := \{0, 1, \dots, N\}$

and $m \in \mathcal{M} := \{H, S, V\}$. The variable n represents the number of packets at the capturing stage, while m indicates whether the server is active in a hardirq processing (in this case, $m = H$), in a softirq processing ($m = S$), i.e., working on capturing packets from the network subsystem to an upper level, or in a vacation period ($m = V$).

Figure 4 shows a state-transition diagram referring to the rates λ , μ_H , μ_S , and μ_V . It reflects how the system proceeds. First of all, if the system is in the empty state $(0, V)$ and a new packet arrives, this causes the beginning of the hardirq. For that reason, the transition from $(0, V)$ to $(1, H)$ happens with rate λ . During the hardirq processing, the system is at states (n, H) . The number of packets can grow with rate λ , except for the state (N, H) because the queue is blocked. When the hardirq ends, due to the task scheduler’s policy, the softirq’s packet-capturing process does not start immediately, but it does after a vacation; therefore, there is a transition from a hardirq state (n, H) to a vacation state (n, V) with rate μ_H .

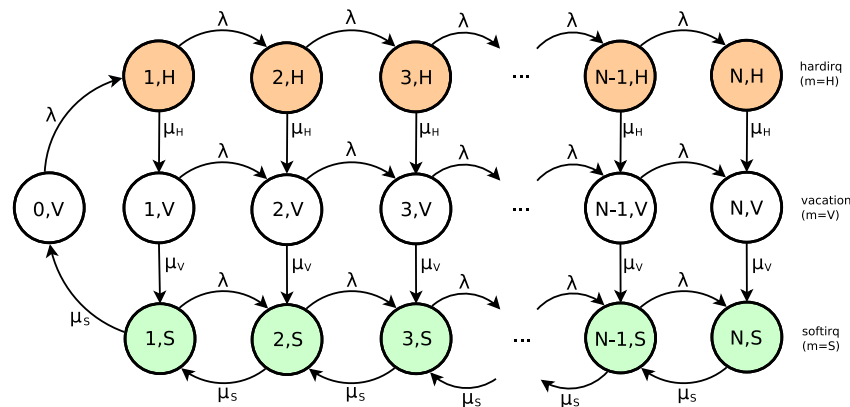


Figure 4. State-transition diagram of model M1.

During the vacation period, at states (n, V) with $1 \leq n \leq N$, the number of packets can also increase with rate λ , except for the blocking state (N, V) . When the system scheduler decides to finish the vacation, there is a transition from a vacation state (n, V) to a softirq state (n, S) with rate μ_V .

At states (n, S) , the processor is capturing packets in the softirq. Then, new packets can arrive with rate λ , except for (N, S) , and packets can leave the capturing system with rate μ_S . The softirq packet-capturing process only ends when the buffer is completely emptied: transition from $(1, S)$ to $(0, V)$ in Figure 4.

3.2. Balance Equations and Steady-State Probabilities

Let $\pi_{n,m}$ be the steady-state probability for the state (n, m) . The balance equations [31] related to the states of Figure 4 can be written as follows.

First, at state $(0, V)$,

$$\lambda\pi_{0,V} = \mu_S\pi_{1,S}. \tag{1}$$

At states $(1, m)$ where $m \in \{H, S, V\}$,

$$(\lambda + \mu_H)\pi_{1,H} = \lambda\pi_{0,V} \tag{2a}$$

$$(\lambda + \mu_V)\pi_{1,V} = \mu_H\pi_{1,H} \tag{2b}$$

$$(\lambda + \mu_S)\pi_{1,S} = \mu_V\pi_{1,V} + \mu_S\pi_{2,S}. \tag{2c}$$

At states (n, m) where $2 \leq n \leq N - 1$ and $m \in \{H, S, V\}$,

$$(\lambda + \mu_H)\pi_{n,H} = \lambda\pi_{n-1,H}, \quad 2 \leq n \leq N - 1 \tag{3a}$$

$$(\lambda + \mu_V)\pi_{n,V} = \lambda\pi_{n-1,V} + \mu_H\pi_{n,H}, \quad 2 \leq n \leq N - 1 \tag{3b}$$

$$(\lambda + \mu_S)\pi_{n,S} = \lambda\pi_{n-1,S} + \mu_V\pi_{n,V} + \mu_S\pi_{n+1,S}, \quad 2 \leq n \leq N - 1. \tag{3c}$$

Finally, at states (N, m) where $m \in \{H, S, V\}$,

$$\mu_H \pi_{N,H} = \lambda \pi_{N-1,H} \tag{4a}$$

$$\mu_V \pi_{N,V} = \lambda \pi_{N-1,V} + \mu_H \pi_{N,H} \tag{4b}$$

$$\mu_S \pi_{N,S} = \lambda \pi_{N-1,S} + \mu_V \pi_{N,V}. \tag{4c}$$

By using the principle of global balance [32], the flow of probability is balanced. If we consider a set of states, the amount of probability flowing in equals the amount flowing out:

$$\lambda(\pi_{n,H} + \pi_{n,V} + \pi_{n,S}) = \mu_S \pi_{n+1,S}, \quad 0 \leq n \leq N - 1. \tag{5}$$

Given that the sum of probabilities must be 1:

$$\sum_{n=0}^N (\pi_{n,H} + \pi_{n,V} + \pi_{n,S}) = 1. \tag{6}$$

After grouping and rewriting Equations (1)–(5), a matrix-geometric solution is reached:

$$\begin{pmatrix} \pi_{n,H} \\ \pi_{n,V} \\ \pi_{n,S} \end{pmatrix} = R^n \begin{pmatrix} \pi_{0,V} \\ 0 \\ 0 \end{pmatrix}, \quad 1 \leq n \leq N - 1,$$

$$\text{where } R = \begin{pmatrix} \frac{\lambda}{\lambda + \mu_H} & 0 & 0 \\ \frac{\lambda \mu_H}{(\lambda + \mu_V)(\lambda + \mu_H)} & \frac{\lambda}{\lambda + \mu_V} & 0 \\ \frac{\lambda}{\mu_S} & \frac{\lambda}{\mu_S} & \frac{\lambda}{\mu_S} \end{pmatrix}. \tag{7a}$$

$$\begin{pmatrix} \pi_{N,H} \\ \pi_{N,V} \\ \pi_{N,S} \end{pmatrix} = Q \begin{pmatrix} \pi_{N-1,H} \\ \pi_{N-1,V} \\ \pi_{N-1,S} \end{pmatrix}, \quad \text{where } Q = \begin{pmatrix} \frac{\lambda}{\mu_H} & 0 & 0 \\ \frac{\lambda}{\mu_V} & \frac{\lambda}{\mu_V} & 0 \\ \frac{\lambda}{\mu_S} & \frac{\lambda}{\mu_S} & \frac{\lambda}{\mu_S} \end{pmatrix}. \tag{7b}$$

After imposing the condition of Equation (6), the value of $\pi_{0,V}$ is obtained as follows:

$$\pi_{0,V} = \frac{1}{K}$$

$$\text{where } K = \begin{pmatrix} 1 & 1 & 1 \end{pmatrix} \left(\sum_{n=0}^{N-1} R^n + QR^{N-1} \right) \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \tag{8}$$

$$\text{and } \sum_{n=0}^{N-1} R^n = (I - R)^{-1} (I - R^N).$$

After computing $\pi_{0,V}$, it is possible to determine every steady-state probability $\pi_{n,m}$ by applying Equations (7a) and (7b).

3.3. Performance Parameters

Once we have the probabilities $\pi_{n,m}$, it is already possible to calculate the performance parameters in terms of blocking probability, throughput, CPU usage, hardirq and softirq frequencies, and softirq mean time.

3.3.1. Blocking Probability (P_B)

It is the probability of dropping packets when the buffer is completely full. This happens at states (N, H) , (N, V) and (N, S) :

$$P_B = \pi_{N,H} + \pi_{N,V} + \pi_{N,S}. \tag{9}$$

3.3.2. Capture Throughput (X_C)

In general, the throughput or completion rate is equal to the number of request completions divided by the time [33]. In our context, capture throughput (X_C) is defined as the mean number of packets processed per second by the queueing system representing the capturing phase of a network-monitoring system. In other words, it is the rate at which the capturing stage is able to transfer packets to its corresponding upper level. It is worth mentioning that the packet departures from the capturing stage take place at states (n, S) where $1 \leq n \leq N$. Capture throughput is also directly related to the packet-arrival rate and the blocking probability:

$$X_C = \mu_S \sum_{n=1}^N \pi_{n,S} = \lambda(1 - P_B). \tag{10}$$

3.3.3. CPU Usage (U_C)

This parameter considers the states in which CPU is devoted to hardirq and softirq processing:

$$U_C = \sum_{n=1}^N \pi_{n,H} + \sum_{n=1}^N \pi_{n,S}. \tag{11}$$

3.3.4. Hardirq Frequency ($f_{hardirq}$)

This parameter measures the number of hardirqs executed per second. The frequency $f_{hardirq}$ is related to the cycle composed of (1) empty buffer time, (2) hardirq time, (3) vacation time with not an empty buffer, and (4) softirq time. We denote T_{cycle} as the cycle mean time and $T_{softirq}$ as the execution mean time of a softirq:

$$T_{cycle} = \frac{1}{\lambda} + \frac{1}{\mu_H} + \frac{1}{\mu_V} + T_{softirq}. \tag{12}$$

Assuming that the packet arrival is a Poisson process, the empty buffer mean time is equal to $1/\lambda$, and we can obtain $f_{hardirq}$ as follows:

$$\begin{aligned} \frac{1}{\lambda} &= T_{cycle} \cdot \pi_{0,V} \\ f_{hardirq} &= \frac{1}{T_{cycle}} = \lambda \cdot \pi_{0,V}. \end{aligned} \tag{13}$$

3.3.5. Softirq Frequency ($f_{softirq}$)

This parameter measures the number of softirqs executed per second. In model M1, there is one softirq for each hardirq. Therefore, both frequencies, $f_{hardirq}$ and $f_{softirq}$, are equal:

$$f_{softirq} = f_{hardirq} = \lambda \cdot \pi_{0,V}. \tag{14}$$

3.3.6. Softirq Mean Time ($T_{softirq}$)

As mentioned previously, it is the mean duration of a softirq. We obtain its expression as follows:

$$T_{softirq} = T_{cycle} \cdot \sum_{n=1}^N \pi_{n,S} = \frac{\sum_{n=1}^N \pi_{n,S}}{f_{softirq}}. \tag{15}$$

4. Analysis for Exponential Time Distributions and an Arbitrary Budget $B < \infty$

The second proposed model, M2, will represent the Linux packet-capturing system with more detail than M1, since it establishes a finite value of the parameter B . Remember that B is the budget, the maximum number of packets that can be processed within a softirq.

It can be said that, while M1 has an exhaustive service discipline during the softirq (this does not end until the buffer is left without any packet), M2 has a limited service discipline during the softirq (this ends when the limit set by B is reached). Thus, the variability of the busy period (the duration of the softirq) is reduced.

The rest of the assumptions coincide with those of the model M1. Thus, we keep the rates λ, μ_H, μ_S and μ_V , as well as buffer size N .

4.1. Markov Chain Related to the Capturing Stage

The analysis of model M2 is based on a Markov chain with a state space $\mathcal{S} := \mathcal{N} \times \mathcal{M}$. The state of the system is $s = (n, m)$ with $n \in \mathcal{N} := \{0, 1, \dots, N\}$ and $m \in \mathcal{M} := \{H, V, 1, 2, \dots, B\}$. As explained in Section 2, n indicates the number of packets in the capturing stage, whereas m allows us to control what type of occupation the server has at any given time.

- $m = H$ if the server is processing a hardirq.
- $m = V$ if the server is on vacation.
- $1 \leq m \leq B$ if the server is processing a softirq and it is serving the m th packet in the current softirq.

The state-transition diagram shown in Figure 5 gathers the procedure of the Linux capturing system. First, if the system is empty, state $(0, V)$, and a packet arrives, this causes the execution of the hardirq service routine. For this reason, in Figure 5, there is a transition for $(0, V)$ to $(1, H)$ with rate λ . During the hardirq, the system remains at states (n, H) . There can be new incoming packets with rate λ , and the system moves to state $(n + 1, H)$. The exception is the state (N, H) , which does not accept new packets, because there is no room in the buffer. When the hardirq finishes, the system moves to a vacation state (n, V) ; this involves a transition from (n, H) to (n, V) with rate μ_H . The vacations of M2 are identical to those of M1; therefore, when the task scheduler determines it, the vacation ends and the execution of a softirq starts. This implies the transition from state (n, V) to state $(n, 1)$ with rate μ_V .

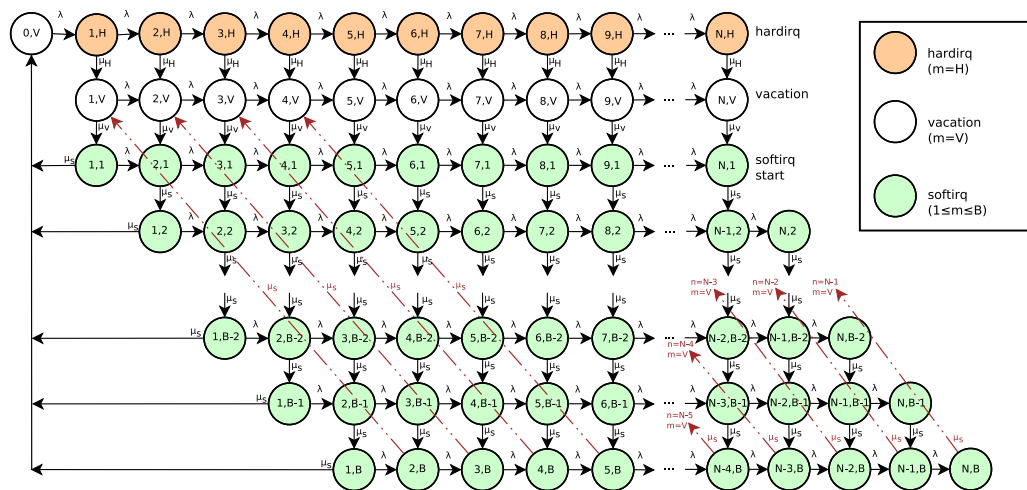


Figure 5. State-transition diagram of model M2.

The execution of a softirq is represented by the set of states (n, m) , where $1 \leq n \leq N$ and $1 \leq m \leq B$. During the softirq, new incoming packets enter the system with rate λ ; therefore, there are transitions from (n, m) to $(n + 1, m)$, except for blocking states (N, m) . It is also possible that the packet departure results in transitions from (n, m) to $(n - 1, m + 1)$ with rate μ_S . Another aspect to consider is the representation of the two options that exist to finish a softirq and start a vacation. One is that the queue is emptied with the departure of the last packet; this corresponds to a transition from state $(1, m)$ to state $(0, V)$ with rate μ_S . The second option is to reach the budget; in this case, there is a transition from state (n, B) to state $(n - 1, V)$ with rate μ_S .

4.2. Balance Equations and Steady-State Probabilities

At steady state, the probabilities $\pi_{n,m}$ satisfy the balance equations of the states in Figure 5. First, at state $(0, V)$,

$$\lambda\pi_{0,V} = \mu_S \sum_{m=1}^B \pi_{1,m}. \tag{16}$$

At states $(1, m)$, where $m \in \{H, S, 1, 2, \dots, B\}$,

$$(\lambda + \mu_H)\pi_{1,H} = \lambda\pi_{0,V} \tag{17a}$$

$$(\lambda + \mu_V)\pi_{1,V} = \mu_H\pi_{1,H} + \mu_S\pi_{2,B} \tag{17b}$$

$$(\lambda + \mu_S)\pi_{1,1} = \mu_V\pi_{1,V} \tag{17c}$$

$$(\lambda + \mu_S)\pi_{1,m} = \mu_S\pi_{2,m-1}, \quad 2 \leq m \leq B. \tag{17d}$$

At states (n, m) , where $2 \leq n \leq N - 1$ and $m \in \{H, S, 1, 2, \dots, B\}$,

$$(\lambda + \mu_H)\pi_{n,H} = \lambda\pi_{n-1,H} \tag{18a}$$

$$(\lambda + \mu_V)\pi_{n,V} = \lambda\pi_{n-1,V} + \mu_H\pi_{n,H} + \mu_S\pi_{n+1,B} \tag{18b}$$

$$(\lambda + \mu_S)\pi_{n,1} = \lambda\pi_{n-1,1} + \mu_V\pi_{n,V} \tag{18c}$$

$$(\lambda + \mu_S)\pi_{n,m} = \lambda\pi_{n-1,m} + \mu_S\pi_{n+1,m-1}, \quad 2 \leq m \leq (B - 1) \tag{18d}$$

$$(\lambda + \mu_S)\pi_{n,B} = \lambda\pi_{n-1,B} + \mu_S\pi_{n+1,B-1}. \tag{18e}$$

Finally, at states (N, m) , where $m \in \{H, S, 1, 2, \dots, B\}$,

$$\mu_H\pi_{N,H} = \lambda\pi_{N-1,H} \tag{19a}$$

$$\mu_V\pi_{N,V} = \lambda\pi_{N-1,V} + \mu_H\pi_{N,H} \tag{19b}$$

$$\mu_S\pi_{N,1} = \lambda\pi_{N-1,1} + \mu_V\pi_{N,V} \tag{19c}$$

$$\mu_S\pi_{N,m} = \lambda\pi_{N-1,m}, \quad 2 \leq m \leq (B - 1) \tag{19d}$$

$$\mu_S\pi_{N,B} = \lambda\pi_{N-1,B}. \tag{19e}$$

In addition, we have the normalization condition:

$$\sum_{n,m} \pi_{n,m} = 1. \tag{20}$$

In order to calculate the steady-state probabilities $\pi_{n,m}$, we perform a matrix development for which we define the following probability vectors of dimension $(B + 2) \times 1$:

$$\vec{\pi}_0 = \begin{pmatrix} \pi_{0,V} \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{pmatrix}, \quad \vec{\pi}_n = \begin{pmatrix} \pi_{n,H} \\ \pi_{n,V} \\ \pi_{n,1} \\ \vdots \\ \pi_{n,B-1} \\ \pi_{n,B} \end{pmatrix}, \quad 1 \leq n \leq N. \tag{21}$$

After performing the corresponding development, we obtain the following matrix relationships between probability vectors. All of them result as a function of the probability $\pi_{0,V}$:

$$\begin{cases} \vec{\pi}_1 = Z_1\vec{\pi}_0 \\ \vec{\pi}_n = Z_n\vec{\pi}_{n-1}, \quad 2 \leq n \leq N. \end{cases} \tag{22}$$

Matrices Z_n are the results of matrix operations involving the rates λ, μ_S, μ_V , and μ_H :

$$\begin{cases} Z_N = -\lambda A^{-1} \\ Z_n = -\lambda(A - \lambda I + BZ_{n+1})^{-1}, \quad 2 \leq n \leq N - 1 \\ Z_1 = -(A - \lambda I + BZ_2)^{-1}C. \end{cases} \tag{23}$$

Matrices A , B , and C , which appear in (23), are square matrices of dimension $(B + 2) \times (B + 2)$. They contain the following elements:

$$A = \begin{pmatrix} -\mu_H & 0 & 0 & 0 & \dots & \dots & 0 \\ \mu_H & -\mu_V & 0 & 0 & \dots & \dots & 0 \\ 0 & \mu_V & -\mu_S & 0 & \dots & \dots & \vdots \\ 0 & 0 & 0 & -\mu_S & \dots & \dots & \vdots \\ \vdots & \vdots & \vdots & 0 & \ddots & \ddots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \ddots & \ddots & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 & -\mu_S \end{pmatrix} \tag{24}$$

$$B = \begin{pmatrix} 0 & 0 & \dots & \dots & \dots & 0 & 0 \\ 0 & 0 & \dots & \dots & \dots & 0 & \mu_S \\ 0 & 0 & 0 & \dots & \dots & 0 & 0 \\ 0 & 0 & \mu_S & 0 & \dots & \vdots & \vdots \\ 0 & 0 & 0 & \mu_S & 0 & \vdots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \dots & \dots & 0 & \mu_S & 0 \end{pmatrix} \tag{25}$$

$$C = \begin{pmatrix} \lambda & 0 & \dots & \dots & 0 & 0 \\ 0 & 0 & \dots & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & & \vdots & \vdots \\ \vdots & \vdots & & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & \dots & 0 & 0 \\ 0 & 0 & \dots & \dots & 0 & 0 \end{pmatrix}. \tag{26}$$

Next, we obtain the value of $\pi_{0,V}$ by replacing, within the normalization condition (20), the sum of $\pi_{n,m}$ with the sum of the equivalent terms extracted from (21)–(23):

$$\pi_{0,V} = \frac{1}{S}; \quad S = 1 + \left[\sum_{n=1}^N \vec{e}_1^T \left(\prod_{i=0}^{n-1} Z_{n-i} \right) \vec{e}_2 \right]. \tag{27}$$

\vec{e}_1 and \vec{e}_2 are vectors of dimension $(B + 2) \times 1$:

$$\begin{aligned} \vec{e}_1^T &= (1 \quad 1 \quad \dots \quad 1), \\ \vec{e}_2^T &= (1 \quad 0 \quad \dots \quad 0). \end{aligned} \tag{28}$$

Once we have the value of $\pi_{0,V}$, we can determine the probabilities $\pi_{n,m}$ by applying (22).

4.3. Performance Parameters

Now, with steady-state probabilities, it is possible to calculate the performance parameters of interest.

4.3.1. Blocking Probability (P_B)

The blocking probability is equal to the sum of the blocking states (N, m) where $m = H$ or $m = V$ or $1 \leq m \leq B$:

$$P_B = \pi_{N,H} + \pi_{N,V} + \sum_{m=1}^B \pi_{N,m}. \tag{29}$$

4.3.2. Capture Throughput (X_C)

The definition of this parameter is the same as that given for model M1. It takes into account the probability of running a softirq and the service rate μ_S :

$$X_C = \mu_S \sum_{n=1}^N \sum_{m=1}^B \pi_{n,m}. \tag{30}$$

4.3.3. CPU Usage (U_C)

This parameter considers the states in which CPU is devoted to hardirq and softirq processing:

$$U_C = \sum_{n=1}^N \pi_{n,H} + \sum_{n=1}^N \sum_{m=1}^B \pi_{n,m}. \tag{31}$$

4.3.4. Hardirq Frequency ($f_{hardirq}$)

This parameter measures, on average, how many times per second the hardirq service routine runs. As was done in model M1, we consider the cycle mean time, T_{cycle} , to calculate $f_{hardirq}$. Here, cycle mean time consists of the empty buffer mean time, the hardirq execution mean time, and the mean time of the set of vacations and softirqs until the buffer is completely emptied:

$$T_{cycle} = \frac{1}{\lambda} + \frac{1}{\mu_H} + k_s \left(\frac{1}{\mu_V} + T_{softirq} \right). \tag{32}$$

$T_{softirq}$ is the softirq mean time and k_s is the mean number of softirqs within the cycle. Since we assume that arrivals follow a Poisson process:

$$\begin{aligned} \frac{1}{\lambda} &= T_{cycle} \cdot \pi_{0,V} \\ f_{hardirq} &= \frac{1}{T_{cycle}} = \lambda \cdot \pi_{0,V}. \end{aligned} \tag{33}$$

4.3.5. Softirq Frequency ($f_{softirq}$)

As in model M1, this parameter indicates the mean number of softirqs executed per second. In model M2, there is always a vacation with $n > 0$ before each softirq. Therefore, we estimate $f_{softirq}$ as follows:

$$\begin{aligned} k_s \cdot \frac{1}{\mu_V} &= T_{cycle} \sum_{n=1}^N \pi_{n,V} \\ f_{softirq} &= \frac{k_s}{T_{cycle}} = \mu_V \sum_{n=1}^N \pi_{n,V}. \end{aligned} \tag{34}$$

4.3.6. Softirq Mean Time ($T_{softirq}$)

We can get the mean duration of a softirq as follows:

$$T_{softirq} = \frac{\sum_{n=1}^N \sum_{m=1}^B \pi_{n,m}}{\mu_V \sum_{n=1}^N \pi_{n,V}}. \tag{35}$$

4.3.7. Mean Number of Packets in a Softirq ($\bar{m}_{softirq}$)

Once we computed the mean time of softirq, we can estimate the mean number of packets processed in a softirq, $\bar{m}_{softirq}$:

$$\bar{m}_{softirq} = \frac{T_{softirq}}{\frac{1}{\mu_S}} = \mu_S T_{softirq} = \frac{\mu_S \sum_{n=1}^N \sum_{m=1}^B \pi_{n,m}}{\mu_V \sum_{n=1}^N \pi_{n,V}}. \tag{36}$$

5. Analysis for General Time Distributions and an Arbitrary Budget $B < \infty$

This section explains the third proposal to represent the Linux packet-capturing stage. We will call it model M3. As with model M2, M3 is also based on a model with vacations and limited service discipline by the budget B of softirq. However, unlike M1 and M2, in this approximation, hardirq and softirq packet-service and vacation times are not exponential, but they follow general distributions. We have the following assumptions for this third model.

- Incoming packets arrive at the system in accordance with a Poisson distribution of rate λ .
- There is only one processor to capture packets.
- Hardirq processing time, H , softirq packet service time, S , and vacation time, V , are all independent variables with distribution functions $H(x)$, $S(x)$, and $V(x)$, respectively. Their corresponding Laplace–Stieltjes transforms are $H^*(\theta)$, $S^*(\theta)$, and $V^*(\theta)$ [34].
- There is a finite budget that we denote as B .
- There is a finite size buffer to store packets before they are processed by the softirq. N is the maximum number of packets in the system. If there are N packets in the system, new, incoming ones are rejected.
- Model M3 follows the procedure that is graphically represented in the general state diagram in Figure 3.

In order to obtain the performance parameters with those assumptions, we first will pose the embedded Markov chain and get its steady-state distribution. Secondly, we will derive the queue length distribution of the continuous-time process. Finally, we will obtain significant performance parameters.

5.1. Embedded Markov Chain

We examine the system at time epochs $\{t_0, t_1, \dots\}$ of hardirq service routine finishing, of vacation termination, or of softirq packet-service completion. The state space of the system is $\mathcal{S} := \{L_i, \delta_i\}$ defined as follows: L_i is the number of packets in the system at the embedded point t_i . If the point t_i is a hardirq finishing instant, then $\delta_i = H$. If the embedded point is a vacation termination instant, then $\delta_i = V$; otherwise, $\delta_i = m$ where $m = 1, 2, \dots, B$ indicates that the point t_i is the packet service completion instant of the m th packet in the present softirq. The state transitions in the process $\{L_i, \delta_i\}$ occur at hardirq finishing instants, vacation termination instants, and packet-departure instants.

Notice that we could have an ambiguity on the embedded point with $(L_i = 1, \delta_i = V)$. It could be the termination point of the empty buffer vacation; i.e., when the first packet arrives at the buffer, this causes the end of the empty buffer vacation and the beginning of a hardirq. However, it can also refer to the end of a regular vacation period with one packet in the buffer; in this case, the system will start with a softirq after the regular vacation. For that reason, we define an embedded point with $(L_i = 1, \delta_i = V_0)$, which corresponds to the empty buffer vacation termination followed by a hardirq, whereas the set of embedded points $(L_i = n, \delta_i = V)$, where $n = 1, 2, \dots, N$, is related to vacation terminations which are followed by a softirq.

When the system is under steady state, the limiting probability distributions $p_{n,m}$ are defined as follows:

$$\begin{aligned}
 p_{n,H} &= \lim_{i \rightarrow \infty} Pr\{L_i = n, \delta_i = H\}, \quad n = 1, 2, \dots, N; \\
 p_{n,V} &= \lim_{i \rightarrow \infty} Pr\{L_i = n, \delta_i = V\}, \quad n = 1, 2, \dots, N; \\
 p_{n,m} &= \lim_{i \rightarrow \infty} Pr\{L_i = n, \delta_i = m\}, \quad n = 0, 1, \dots, N - 1; \quad m = 1, 2, \dots, B; \\
 p_{1,V0} &= \lim_{i \rightarrow \infty} Pr\{L_i = 1, \delta_i = V_0\}.
 \end{aligned}$$

Taking into account that packets arrive at the system according to a Poisson process, we also define the probabilities $u_i, v_i,$ and w_i .

u_i is the probability that i packets arrive during a hardirq time H :

$$u_i = \int_0^\infty e^{-\lambda x} \frac{(\lambda x)^i}{i!} dH(x). \tag{37}$$

v_i is the probability that i packets arrive during a vacation time V :

$$v_i = \int_0^\infty e^{-\lambda x} \frac{(\lambda x)^i}{i!} dV(x). \tag{38}$$

w_i is the probability that i packets arrive during a softirq packet processing time S :

$$w_i = \int_0^\infty e^{-\lambda x} \frac{(\lambda x)^i}{i!} dS(x). \tag{39}$$

We also define the probabilities $u_k^c, v_k^c,$ and w_k^c :

$$u_k^c = \sum_{i=k}^\infty u_i, \quad v_k^c = \sum_{i=k}^\infty v_i, \quad w_k^c = \sum_{i=k}^\infty w_i. \tag{40}$$

If $H^{*(i)}, V^{*(i)}$ and $S^{*(i)}$ denote the i th derivative of H^*, V^* and S^* , then

$$u_i = \left[\frac{(-\lambda)^i}{i!} \right] H^{*(i)}(\lambda), \quad v_i = \left[\frac{(-\lambda)^i}{i!} \right] V^{*(i)}(\lambda), \quad w_i = \left[\frac{(-\lambda)^i}{i!} \right] S^{*(i)}(\lambda), \tag{41}$$

where $H^*(\theta), V^*(\theta)$ and $S^*(\theta)$ are the Laplace–Stieltjes transforms of $H(x), V(x)$ and $S(x)$, respectively.

All those probabilities satisfy a set of equations related to the embedded Markov chain. Below, we state those equations by separating them into different subsets.

At hardirq service routine termination points,

$$p_{n,H} = p_{1,V0} u_{n-1}, \quad 0 \leq n \leq N - 1 \tag{42a}$$

$$p_{N,H} = p_{1,V0} u_{N-1}^C. \tag{42b}$$

At vacation termination points that are followed by a softirq,

$$p_{n,V} = \sum_{k=1}^n v_{n-k} (p_{k,H} + p_{k,B}), \quad 1 \leq n \leq N \tag{43a}$$

$$p_{N,V} = \sum_{k=1}^{N-1} v_{N-k}^C (p_{k,H} + p_{k,B}) + p_{N,H}. \tag{43b}$$

At softirq packet service completion points,

$$p_{n,1} = \sum_{k=1}^{n+1} w_{n-k+1} p_{k,V}, \quad 0 \leq n \leq N - 2 \tag{44a}$$

$$p_{N-1,1} = \sum_{k=1}^N w_{N-k}^C p_{k,V} \tag{44b}$$

$$p_{n,m} = \sum_{k=1}^{n+1} w_{n-k+1} p_{k,m-1}, \quad 0 \leq n \leq N-2, \quad 2 \leq m \leq B \tag{44c}$$

$$p_{N-1,m} = \sum_{k=1}^{N-1} w_{N-k}^C p_{k,m-1}, \quad 2 \leq m \leq B. \tag{44d}$$

Finally, at the empty buffer vacation termination point,

$$p_{1,V0} = \sum_{m=1}^B p_{0,m}. \tag{45}$$

We also have the equation of the sum of probabilities:

$$\sum_{n=1}^N p_{n,H} + \sum_{n=1}^N p_{n,V} + \sum_{n=0}^{N-1} \sum_{m=1}^B p_{n,m} + p_{1,V0} = 1. \tag{46}$$

The next step is to solve the probabilities $p_{n,m}$ of the embedded Markov chain under steady state. To do this, we rewrite Equations (42)–(46) in matrix form as a function of coefficients $u_i, v_i, w_i, u_i^c, v_i^c,$ and w_i^c . Applying an analogous method to that used to calculate the steady-state probabilities of model M2 (see Section 4.2), we obtain the probabilities $p_{n,m}$.

Once the probabilities $p_{n,m}$ are obtained, we establish some system parameters that will be used later to estimate some performance parameters. First, we define b_0 as the probability of an embedded point to be a vacation termination point prior to a hardirq. Secondly, we define b_1 as the probability of an embedded point to be a hardirq termination point. Thirdly, we define b_2 as the probability of an embedded point to be a vacation termination point prior to a softirq. Lastly, we define b_3 as the probability of an embedded point to be a packet service completion point. Thus, we have

$$b_0 = p_{1,V0} \quad ; \quad b_1 = \sum_{n=1}^N p_{n,H} \quad ; \quad b_2 = \sum_{n=1}^N p_{n,V} \quad ; \quad b_3 = \sum_{n=0}^{N-1} \sum_{m=1}^B p_{n,m}. \tag{47}$$

Another parameter of interest is the frequency of embedded points. We denote it by σ . The inverse of σ is interpreted as the average interval between consecutive embedded points and it can be expressed by

$$\sigma^{-1} = b_0 \frac{1}{\lambda} + b_1 E(H) + b_2 E(V) + b_3 E(S). \tag{48}$$

5.2. General Queue Length Distribution

In this subsection, we develop the calculation of the probability distribution of our queue M/G/1/N with hardirq service routines, vacations, and limited service discipline for softirq packet processing.

Let variable L be the number of packets in the steady state of the system at an arbitrary instant of time ($0 \leq L \leq N$). Let variable δ be the system state from the processor’s point of view. Thus, if $\delta = H$, the processor is attending a hardirq service routine. If $\delta = V$, the processor is on vacation with $L > 0$. If $\delta = V_0$, the processor is also on vacation, but the queue is empty ($L = 0$) and the vacation will finish when the first new packet arrives at the system. Finally, if $\delta = m$ being $1 \leq m \leq B$, it denotes that the processor is serving the m th packet in the present softirq.

The general queue length distribution of the time continuous process can be treated by the supplementary variable technique [26]. For that reason, we consider the following supplementary variables:

- \hat{H} = forward recurrence hardirq service time, i.e., the remaining service time for the hardirq;
- \hat{V} = forward recurrence vacation time, i.e., the remaining vacation time for the processor on vacation, not being the buffer empty ($n > 0$);
- \hat{V}_0 = forward recurrence vacation time when there is not any packet in the buffer ($n = 0$), i.e., the remaining vacation time until the arrival of the first packet; due to the fact that packet arrivals follow a Poisson process, \hat{V}_0 has a exponential distribution; and
- \hat{S} = forward recurrence softirq packet service time, i.e., the remaining service time for the packet departure in the current softirq execution.

We also define the following backward recurrence service times:

- \tilde{H} = backward recurrence hardirq’s service time, i.e., the elapsed service time for the hardirq;
- \tilde{V} = backward recurrence vacation time, i.e., the elapsed vacation time for the processor on vacation, not being the buffer empty ($n > 0$);
- \tilde{V}_0 = backward recurrence vacation time, i.e., the elapsed vacation time for the processor on vacation with empty buffer ($n = 0$); and
- \tilde{S} = backward recurrence softirq packet service time, i.e., the elapsed service time for the packet that is being served in softirq.

We intend to determine the expressions of probabilities $\pi_{n,H}$, $\pi_{n,V}$ and $\pi_{n,m}$, starting from

$$\pi_{n,H}(\tau)d\tau = Pr\{L = n, \delta = H, \tau < \hat{H} \leq \tau + d\tau\}, \quad 1 \leq n \leq N \tag{49a}$$

$$\pi_{0,V}(\tau)d\tau = Pr\{L = 0, \delta = V_0, \tau < \hat{V}_0 \leq \tau + d\tau\} \tag{49b}$$

$$\pi_{n,V}(\tau)d\tau = Pr\{L = n, \delta = V, \tau < \hat{V} \leq \tau + d\tau\}, \quad 1 \leq n \leq N \tag{49c}$$

$$\pi_{n,m}(\tau)d\tau = Pr\{L = n, \delta = m, \tau < \hat{S} \leq \tau + d\tau\}, \quad 1 \leq n \leq N, 1 \leq m \leq B, \tag{49d}$$

and knowing that their corresponding Laplace transforms are

$$\pi_{n,H}^*(\theta) = \int_0^\infty e^{-\theta\tau} \pi_{n,H}(\tau)d\tau, \quad 1 \leq n \leq N \tag{50a}$$

$$\pi_{n,V}^*(\theta) = \int_0^\infty e^{-\theta\tau} \pi_{n,V}(\tau)d\tau, \quad 0 \leq n \leq N \tag{50b}$$

$$\pi_{n,m}^*(\theta) = \int_0^\infty e^{-\theta\tau} \pi_{n,m}(\tau)d\tau, \quad 1 \leq n \leq N, 1 \leq m \leq B. \tag{50c}$$

The integrals in Laplace–Stieltjes transforms (50a)–(50c) can be evaluated by conditioning on $\chi(\tilde{H})$, $\chi(\tilde{V})$ and $\chi(\tilde{S})$, the number of packets arrived during \tilde{H} , the backward recurrence hardirq service time, \tilde{V} , the backward recurrence vacation time, and \tilde{S} , the backward recurrence softirq packet service time. Thus, we have the following cases.

- Probabilities related to the hardirq service routine include

$$\pi_{n,H}^*(\theta) = Pr\{\delta = H\} \cdot E\left[e^{-\theta\hat{H}}|\chi(\tilde{H} = n - 1)\right] \cdot Pr\{\chi(\tilde{H}) = n - 1\}, \tag{51a}$$

$$1 \leq n \leq N - 1$$

$$\pi_{N,H}^*(\theta) = Pr\{\delta = H\} \sum_{k=N-1}^\infty E\left[e^{-\theta\hat{H}}|\chi(\tilde{H} = k)\right] Pr\{\chi(\tilde{H}) = k\}. \tag{51b}$$

- Probabilities related to vacations include

$$\pi_{0,V}^*(\theta) = Pr\{\delta = V_0\} \frac{\sum_{m=1}^B p_{0,m} E[e^{-\theta \hat{V}_0} | \chi(\tilde{V}_0 = 0)] Pr\{\chi(\tilde{V}_0) = 0\}}{\sum_{m=1}^B p_{0,m}} \tag{52a}$$

$$\pi_{n,V}^*(\theta) = Pr\{\delta = V\} \cdot \frac{\sum_{j=1}^n (p_{j,H} + p_{j,B}) E[e^{-\theta \hat{V}} | \chi(\tilde{V} = n - j)] Pr\{\chi(\tilde{V}) = n - j\}}{\sum_{j=1}^N p_{j,H} + \sum_{j=1}^{N-1} p_{j,B}}, \tag{52b}$$

1 ≤ n ≤ N

$$\pi_{N,V}^*(\theta) = Pr\{\delta = V\} \left(\frac{\sum_{j=1}^{N-1} (p_{j,H} + p_{j,B}) \sum_{k=N-j}^{\infty} E[e^{-\theta \hat{V}} | \chi(\tilde{V} = k)] Pr\{\chi(\tilde{V}) = k\}}{\sum_{j=1}^N p_{j,H} + \sum_{j=1}^{N-1} p_{j,B}} + \frac{p_{N,H} \sum_{k=0}^{\infty} E[e^{-\theta \hat{V}} | \chi(\tilde{V} = k)] Pr\{\chi(\tilde{V}) = k\}}{\sum_{j=1}^N p_{j,H} + \sum_{j=1}^{N-1} p_{j,B}} \right). \tag{52c}$$

- Probabilities related to softirq packet processing include

$$\pi_{n,1}^*(\theta) = Pr\{\delta = 1\} \frac{\sum_{j=1}^n p_{j,V} E[e^{-\theta \hat{S}} | \chi(\tilde{S} = n - j)] Pr\{\chi(\tilde{S}) = n - j\}}{\sum_{j=1}^N p_{j,V}}, \tag{53a}$$

1 ≤ n ≤ N - 1

$$\pi_{N,1}^*(\theta) = Pr\{\delta = 1\} \frac{\sum_{j=1}^N p_{j,V} \sum_{k=N-j}^{\infty} E[e^{-\theta \hat{S}} | \chi(\tilde{S} = k)] Pr\{\chi(\tilde{S}) = k\}}{\sum_{j=1}^N p_{j,V}} \tag{53b}$$

$$\pi_{n,m}^*(\theta) = Pr\{\delta = m\} \frac{\sum_{j=1}^n p_{j,m-1} E[e^{-\theta \hat{S}} | \chi(\tilde{S} = n - j)] Pr\{\chi(\tilde{S}) = n - j\}}{\sum_{j=1}^{N-1} p_{j,m-1}}, \tag{53c}$$

1 ≤ n ≤ N - 1, 2 ≤ m ≤ B

$$\pi_{N,m}^*(\theta) = Pr\{\delta = m\} \frac{\sum_{j=1}^{N-1} p_{j,m-1} \sum_{k=N-j}^{\infty} E[e^{-\theta \hat{S}} | \chi(\tilde{S} = k)] Pr\{\chi(\tilde{S}) = k\}}{\sum_{j=1}^{N-1} p_{j,m-1}}, \tag{53d}$$

2 ≤ m ≤ B.

Probabilities $Pr\{\delta\}$ can be expressed as a function of the embedded Markov chain probabilities, the average durations of the different types of intervals (hardirq, vacation, or softirq packet service) and the parameter σ^{-1} (the average interval between consecutive embedded points):

$$Pr\{\delta = V_0\} = \frac{\frac{1}{\lambda} p_{1,V_0}}{\sigma^{-1}} = \frac{\sigma}{\lambda} p_{1,V_0} = \frac{\sigma}{\lambda} \sum_{m=1}^B p_{0,m} \tag{54a}$$

$$Pr\{\delta = H\} = \frac{E[H] \sum_{n=1}^N p_{n,H}}{\sigma^{-1}} = \sigma E[H] \sum_{n=1}^N p_{n,H} \tag{54b}$$

$$Pr\{\delta = V\} = \frac{E[V] \sum_{n=1}^N p_{n,V}}{\sigma^{-1}} = \sigma E[V] \sum_{n=1}^N p_{n,V} \tag{54c}$$

$$Pr\{\delta = m\} = \frac{E[S] \sum_{n=0}^{N-1} p_{n,m}}{\sigma^{-1}} = \sigma E[S] \sum_{n=0}^{N-1} p_{n,m}. \tag{54d}$$

After performing algebraic manipulation, we get the following expressions:

$$\pi_{0,V}^*(\theta) = \frac{\sigma}{\lambda} p_{1,V_0} \left(\frac{\lambda}{\lambda + \theta} \right) \left(\frac{\lambda}{\lambda - \theta} \right) \tag{55a}$$

$$\pi_{n,H}^*(\theta) = \frac{\sigma}{\lambda} \left\{ H^*(\theta) \left(\sum_{j=1}^N p_{j,H} \right) \left(\frac{\lambda}{\lambda - \theta} \right)^n - \sum_{j=1}^n p_{j,H} \left(\frac{\lambda}{\lambda - \theta} \right)^{n-j+1} \right\}, \quad 1 \leq n \leq N - 1 \tag{55b}$$

$$\pi_{N,H}^*(\theta) = \frac{-\sigma}{\theta} \left\{ H^*(\theta) \left(\sum_{j=1}^N p_{j,H} \right) \left(\frac{\lambda}{\lambda - \theta} \right)^{N-1} - \sum_{j=1}^N p_{j,H} \left(\frac{\lambda}{\lambda - \theta} \right)^{N-j} \right\} \tag{55c}$$

$$\pi_{n,V}^*(\theta) = \frac{\sigma}{\lambda} \left\{ V^*(\theta) \sum_{j=1}^n (p_{j,H} + p_{j,B}) \left(\frac{\lambda}{\lambda - \theta} \right)^{n-j+1} - \sum_{j=1}^n p_{j,V} \left(\frac{\lambda}{\lambda - \theta} \right)^{n-j+1} \right\}, \quad 1 \leq n \leq N - 1 \tag{55d}$$

$$\pi_{N,V}^*(\theta) = \frac{-\sigma}{\theta} \left\{ V^*(\theta) \sum_{j=1}^N p_{j,H} \left(\frac{\lambda}{\lambda - \theta} \right)^{N-j} + V^*(\theta) \sum_{j=1}^{N-1} p_{j,B} \left(\frac{\lambda}{\lambda - \theta} \right)^{N-j} - \sum_{j=1}^N p_{j,V} \left(\frac{\lambda}{\lambda - \theta} \right)^{N-j} \right\} \tag{55e}$$

$$\pi_{n,1}^*(\theta) = \frac{\sigma}{\lambda} \left\{ S^*(\theta) \sum_{j=1}^n p_{j,V} \left(\frac{\lambda}{\lambda - \theta} \right)^{n-j+1} - \sum_{j=0}^{n-1} p_{j,1} \left(\frac{\lambda}{\lambda - \theta} \right)^{n-j} \right\}, \quad 1 \leq n \leq N - 1 \tag{55f}$$

$$\pi_{N,1}^*(\theta) = \frac{-\sigma}{\theta} \left\{ S^*(\theta) \sum_{j=1}^N p_{j,V} \left(\frac{\lambda}{\lambda - \theta} \right)^{N-j} - \sum_{j=0}^{N-1} p_{j,1} \left(\frac{\lambda}{\lambda - \theta} \right)^{N-j-1} \right\} \tag{55g}$$

$$\pi_{n,m}^*(\theta) = \frac{\sigma}{\lambda} \left\{ S^*(\theta) \sum_{j=1}^n p_{j,m-1} \left(\frac{\lambda}{\lambda - \theta} \right)^{n-j+1} - \sum_{j=0}^{n-1} p_{j,m} \left(\frac{\lambda}{\lambda - \theta} \right)^{n-j} \right\}, \quad 1 \leq n \leq N - 1, 2 \leq m \leq B \tag{55h}$$

$$\pi_{N,m}^*(\theta) = \frac{-\sigma}{\theta} \left\{ S^*(\theta) \sum_{j=1}^{N-1} p_{j,m-1} \left(\frac{\lambda}{\lambda - \theta} \right)^{N-j} - \sum_{j=0}^{N-1} p_{j,m} \left(\frac{\lambda}{\lambda - \theta} \right)^{N-j-1} \right\}, \quad 2 \leq m \leq B. \tag{55i}$$

Since σ^{-1} is the average interval between consecutive embedded points, σ/λ in above equations is interpreted as the inverse of the average number of arrivals between embedded points.

Finally, we obtain the queue length general distribution by introducing $\theta = 0$ in the expressions of $\pi_{n,m}^*(\theta)$, where $n \in \{0, 1, \dots, N\}$ and $m \in \{H, V, 1, 2, \dots, B\}$, in Equations (55a)–(55i), i.e., $\pi_{n,m} = \pi_{n,m}^*(\theta = 0)$. We have

$$\pi_{0,V} = \frac{\sigma}{\lambda} p_{1,V0} \tag{56a}$$

$$\pi_{n,H} = \frac{\sigma}{\lambda} \left\{ p_{1,V0} - \sum_{j=1}^n p_{j,H} \right\}, \quad 1 \leq n \leq N - 1 \tag{56b}$$

$$\pi_{N,H} = \frac{\sigma}{\lambda} \left\{ \lambda E(H) p_{1,V0} - (N - 1) p_{1,V0} + \sum_{j=1}^N (N - j) p_{j,H} \right\} \tag{56c}$$

$$\pi_{n,V} = \frac{\sigma}{\lambda} \left\{ \sum_{j=1}^n (p_{j,H} + p_{j,B}) - \sum_{j=1}^n p_{j,V} \right\}, \quad 1 \leq n \leq N - 1 \tag{56d}$$

$$\pi_{N,V} = \frac{\sigma}{\lambda} \left\{ \lambda E(V) \sum_{j=1}^N (p_{j,H} + p_{j,B}) - \sum_{j=1}^N (p_{j,H} + p_{j,B})(N - j) + \sum_{j=1}^N p_{j,V}(N - j) \right\} \tag{56e}$$

$$\pi_{n,1} = \frac{\sigma}{\lambda} \left\{ \sum_{j=1}^n p_{j,V} - \sum_{j=0}^{n-1} p_{j,1} \right\}, \quad 1 \leq n \leq N - 1 \tag{56f}$$

$$\pi_{N,1} = \frac{\sigma}{\lambda} \left\{ \lambda E(S) \sum_{j=1}^N p_{j,V} - \sum_{j=1}^N p_{j,V}(N-j) + \sum_{j=0}^{N-1} p_{j,1}(N-j-1) \right\} \tag{56g}$$

$$\pi_{n,m} = \frac{\sigma}{\lambda} \left\{ \sum_{j=1}^n p_{j,m-1} - \sum_{j=0}^{n-1} p_{j,m} \right\}, \quad 1 \leq n \leq N-1, \quad 2 \leq m \leq B \tag{56h}$$

$$\pi_{N,m} = \frac{\sigma}{\lambda} \left\{ \lambda E(S) \sum_{j=1}^{N-1} p_{j,m-1} - \sum_{j=1}^{N-1} (N-j)p_{j,m-1} + \sum_{j=0}^{N-1} (N-j-1)p_{j,m} \right\}, \tag{56i}$$

$$2 \leq m \leq B.$$

5.3. Performance Parameters

After calculating the probabilities of the embedded Markov chain and the joint distribution of the queue length, we can determine the performance parameters of model M3 like blocking probability, capture throughput, hardirq frequency, softirq frequency, softirq mean time, and mean number of packets in a softirq.

5.3.1. Blocking Probability (P_B)

The blocking probability of model M3 can be calculated with the same formula as model M2, that is, by adding the probabilities of blocking states (N, m) where $m \in \{H, V, 1, 2, \dots, B\}$:

$$P_B = \pi_{N,H} + \pi_{N,V} + \sum_{m=1}^B \pi_{N,m}. \tag{57}$$

We can also obtain the blocking probability with the results derived from the analysis of the embedded Markov chain. If we consider softirq as the main activity of our model, since we have packet departures as a result of softirq processing, we can define $\rho = \lambda E(S)$ as the offered load. We can also define the carried load ρ' as the fraction of the time that the softirq is busy:

$$\rho' = \frac{b_3 E(S)}{b_1 E(H) + b_2 E(V) + b_3 E(S) + b_0 \frac{1}{\lambda}} = \sigma b_3 E(S). \tag{58}$$

The blocking probability P_B is also given by

$$P_B = \frac{(\rho - \rho')}{\rho} = 1 - \frac{\rho'}{\rho} = 1 - \frac{\sigma b_3}{\lambda} = 1 - \frac{\sigma}{\lambda} \sum_{n=0}^{N-1} \sum_{m=1}^B p_{n,m}. \tag{59}$$

We have verified that both expressions we have obtained for the blocking probability P_B are equivalent. If we introduce the expressions (56c), (56e) and (56i) of $\pi_{N,H}$, $\pi_{N,V}$ and $\pi_{N,m}$, respectively, in the expression (57) of P_B , the result is the same as that of expression (59) of P_B .

5.3.2. Capture Throughput (X_C)

We calculate the capture throughput of model M3 with λ and the blocking probability P_B computed with (57) or (59):

$$X_C = \lambda(1 - P_B). \tag{60}$$

5.3.3. CPU Usage (U_C)

This parameter is estimated with the steady-state probabilities of hardirq and softirq, by applying the same expression as that of model M2:

$$U_C = \sum_{n=1}^N \pi_{n,H} + \sum_{n=1}^N \sum_{m=1}^B \pi_{n,m}. \tag{61}$$

5.3.4. Hardirq Frequency ($f_{hardirq}$)

As in model M2, we apply the same concepts of cycle mean time T_{cycle} , softirq mean time $T_{softirq}$, and Poisson interarrival times, we get the frequency of hardirq $f_{hardirq}$:

$$f_{hardirq} = \frac{1}{T_{cycle}} = \lambda \cdot \pi_{0,V}. \tag{62}$$

5.3.5. Softirq Frequency ($f_{softirq}$)

With the same reasoning used for model M2, this is the expression of softirq frequency $f_{softirq}$ for model M3:

$$f_{softirq} = \frac{\sum_{n=1}^N \pi_{n,V}}{E(V)}. \tag{63}$$

5.3.6. Softirq Mean Time ($T_{softirq}$)

We can get the mean time of a softirq of model M3 as

$$T_{softirq} = \frac{T_{cycle} \sum_{n=1}^N \sum_{m=1}^B \pi_{n,m}}{k_s} = \frac{E(V) \sum_{n=1}^N \sum_{m=1}^B \pi_{n,m}}{\sum_{n=1}^N \pi_{n,V}}. \tag{64}$$

5.3.7. Mean Number of Packets in a Softirq ($\bar{m}_{softirq}$)

Finally, we compute the mean number of packets processed in a softirq as

$$\bar{m}_{softirq} = \frac{T_{softirq}}{E(S)} = \frac{E(V) \sum_{n=1}^N \sum_{m=1}^B \pi_{n,m}}{E(S) \sum_{n=1}^N \pi_{n,V}}. \tag{65}$$

It is also possible to estimate $\bar{m}_{softirq}$ by using the probabilities of the embedded Markov chain:

$$\bar{m}_{softirq} = \frac{\sum_{m=1}^B m p_{0,m} + B \sum_{n=1}^{N-1} p_{n,B}}{\sum_{m=1}^B p_{0,m} + \sum_{n=1}^{N-1} p_{n,B}}. \tag{66}$$

6. Model Evaluation

This section shows model evaluation results. We obtain the analytical curves derived from models M1, M2, and M3 by implementing them in MATLAB. In addition, in order to carry out some comparisons, we present some experimental values obtained from a real Linux-based packet-capture and analysis probe [35] that operates within a laboratory measurement platform [36]. For comparing both types of results, it is necessary to have some input parameters of the models (for instance, λ , $1/\mu_H$, $1/\mu_S$, $1/\mu_V$) whose values come from measurements at the experimental platform.

6.1. Evaluation Scenarios

We have three evaluation scenarios: V1, V2, and V3. Each of them is characterized by having different vacation behavior as shown in Figure 6. Specifically, Figure 6 represents the mean time of vacation normalized with respect to the maximum duration of a softirq, $\bar{E}(V) = (1/\mu_V)/(B/\mu_S)$, for different input packet rates normalized with respect to the softirq packet service rate $\bar{\lambda} = \lambda/\mu_S$. We use $B = 300$ because this is the typical value of budget. In Figure 6, it can be seen that $\bar{E}(V)$ does not remain constant. This is because

the values of $\bar{E}(V)$ are extracted from the real probe where analysis operations after the capture require higher computational consumptions as the incoming packet rate increases. Thus, V1 corresponds to a low analysis load scenario, V2 to an intermediate analysis load scenario, and V3 to a high analysis load scenario. The higher the analysis load is, the more time is required for the tasks that the processor should perform on vacation.

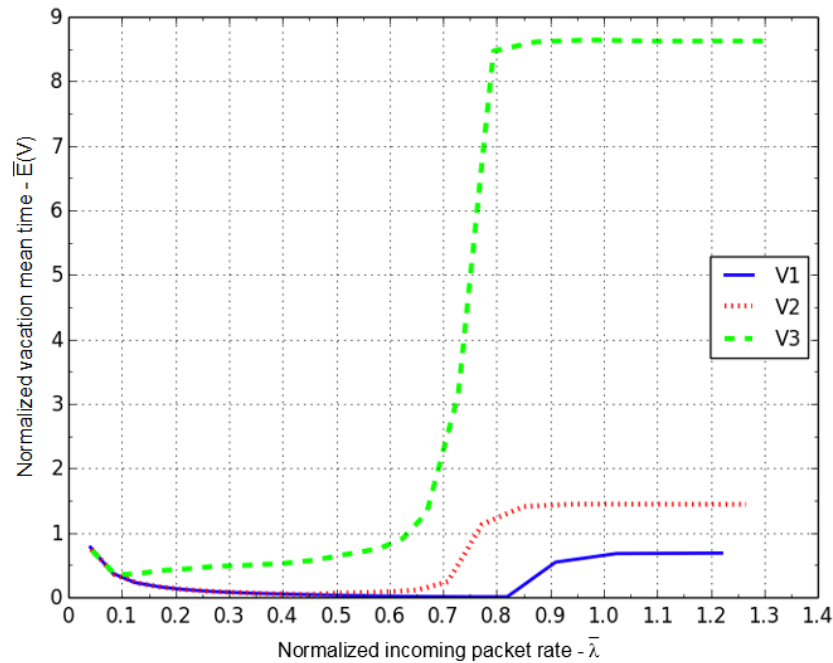


Figure 6. Evaluation scenarios: V1, V2, and V3.

6.2. Performance Results

As mentioned before, we have performance results of the analytical models M1, M2, M3 as well as measurements of the real packet-capture probe (we refer to them as “Lab” in the figures of this subsection). Below, we present some examples of them.

First, Figure 7 shows how the normalized capture throughput $\bar{X}_C = X_C/\mu_S$ varies with the different scenarios. We evaluate it with $N = 200$ and $B = 300$ (same values as the settings of the network driver of the probe). We observe that model M1 has a similar behavior for scenarios V1, V2, and V3; that is, it is practically unaffected by the different vacation times. This is due to the exhaustive discipline that has been assumed for the capture process, which allows us to extend indefinitely the time spent on the capture. In this way, the maximum throughput is obtained in saturation, $\bar{X}_C \approx 1$, at the expense of eliminating, in practice, the periods of vacation. Model M1 begins to saturate from the value $\bar{\lambda} \approx 0.9$. If we compare M1 with laboratory data, it only fits for values below $\bar{\lambda} \approx 0.8$.

On the other hand, unlike M1, model M2 is really affected by the inactivity periods and, consequently, the capture throughput decreases. The case of model M2 with scenario V1 (M2_V1) is very close to the values of the real probe (Lab). The maximum is reached with $\bar{X}_C \approx 0.8$. The subsequent decrease is due to the increase in vacation times and the termination of softirq by budget. For the highest rates, $\bar{\lambda} > 1$, throughput becomes approximately constant because the mean time of vacation remains stable and the duration of the capture process is set by the limit B . Cases M2_V2 and M2_V3 allow us to predict the system behavior in other scenarios. We can see that the shape is similar to M2_V1, but throughput values are lower.

Secondly, Figure 8 exhibits the variation of throughput with respect to budget B on scenario V1. We keep $N = 200$ and the different values of B are $B \rightarrow \infty$ (M1 in Figure 8), $B = 300$, $B = 200$, and $B = 100$. Case M2_B300 fits the laboratory values. If the budget is lower, cases M2_B200, and M2_B100, we observe that the throughput decreases in

the saturation zone ($\bar{\lambda} > 0.8$). In addition, the lower B , the more throughput decreases. This is because when budget is exhausted, the time devoted to capture decreases with B . If we evaluate model M2 for values $B > 300$, we obtain throughputs greater than in case M2_B300, but always below the extreme case M1.

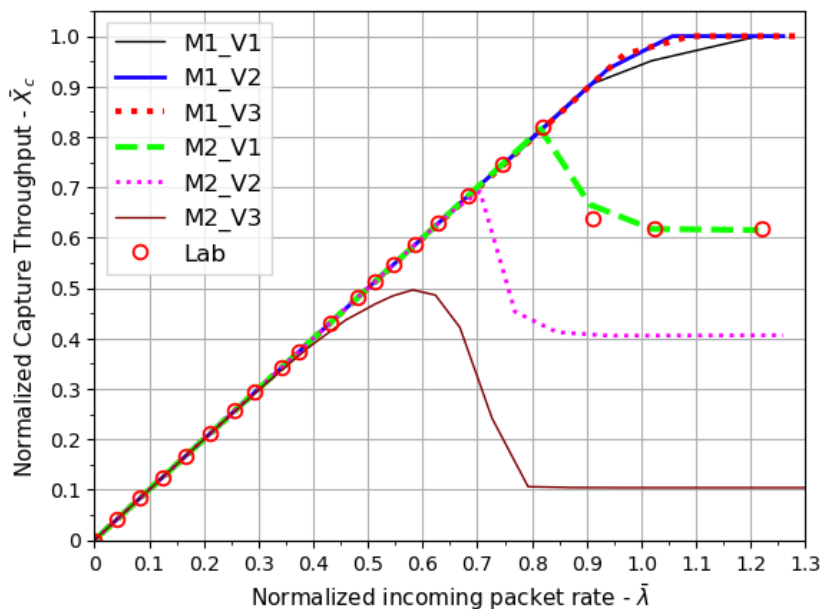


Figure 7. Results of normalized capture throughput for different scenarios (V1, V2, and V3).

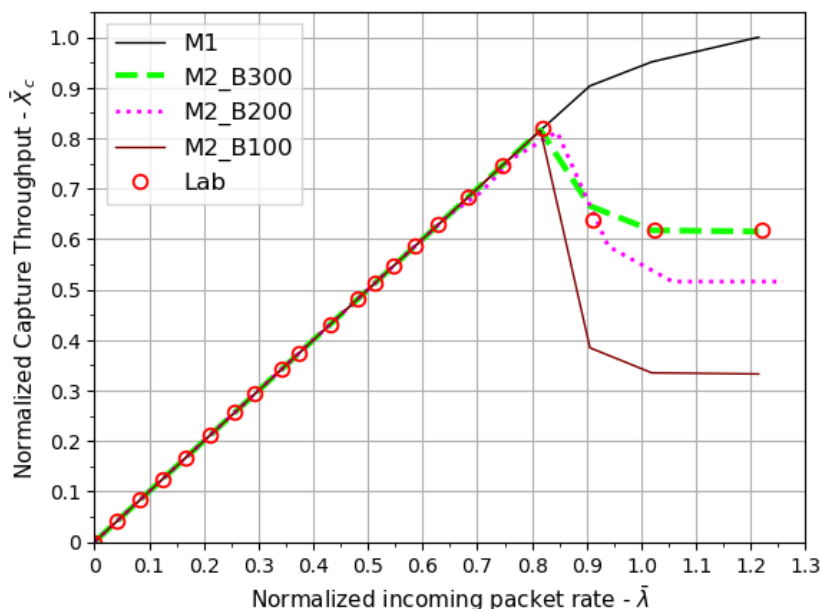


Figure 8. Results of normalized capture throughput for different budgets and scenario V1.

Next, Figure 9 represents the variation of softirq frequency. This parameter can give us an idea of the number of context switches between the capture and vacation periods. We distinguish three zones for scenarios V1, V2, and V3. With low rates, as the input rate increases, the number of softirq per second grows until it reaches a peak. From there, with intermediate input rates, due to the increase in vacation times, the frequency of softirq decreases. Finally, there is a third zone for higher input rates where, in the case of model M1, the softirq frequency decreases because the duration of softirq is prolonged indefinitely;

there are practically no vacations and $f_{softirq} \rightarrow 0$. On the contrary, in the third zone of models with finite budget, the softirq frequency remains constant with a low value. This is due to the fact that, in that third zone, the softirq mean time takes the value B/μ_S in M2 and $B \cdot E(S)$ in M3. Once again, as can be seen in Figure 9, the case of model M2 with scenario V1 fits the laboratory measurement.

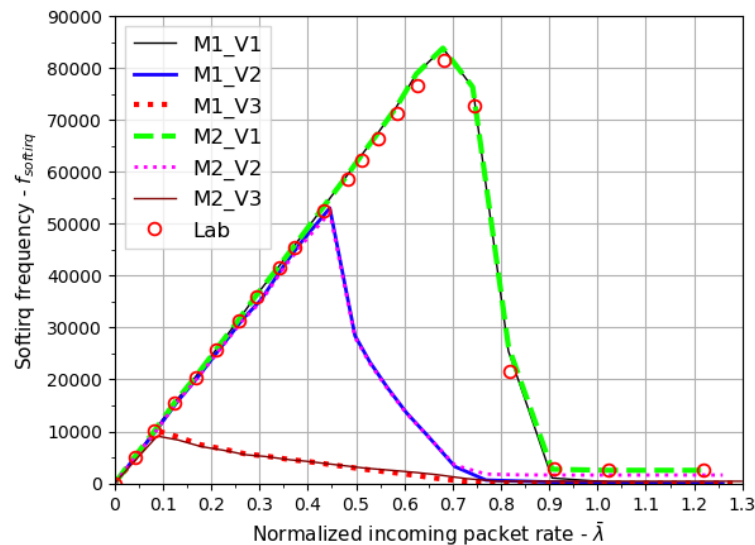


Figure 9. Results of softirq frequency for different scenarios (V1, V2, and V3).

We have also analyzed results of hardirq frequency in models with finite budget. We observe that, for input rates where budget is not reached, $f_{hardirq} \approx f_{softirq}$ and, for input rates in the saturation zone, there is an extreme value such as $f_{hardirq} \rightarrow 0$.

Finally, Figure 10 shows the mean number of packets per softirq $\bar{m}_{softirq}$. It allows us to see from what input rate, the execution of the softirq exhausts the value of budget ($B = 300$ in the graph). Obviously, scenario V3 exhausts the budget with the lowest input rate, and scenario V1 reaches the budget with the highest rate. Again, the case M2_V1 fits the values measured in the laboratory probe.

It is worth mentioning that we have also evaluated models with buffer sizes greater than 200 (specifically, $N = 512$). The obtained results have been similar.

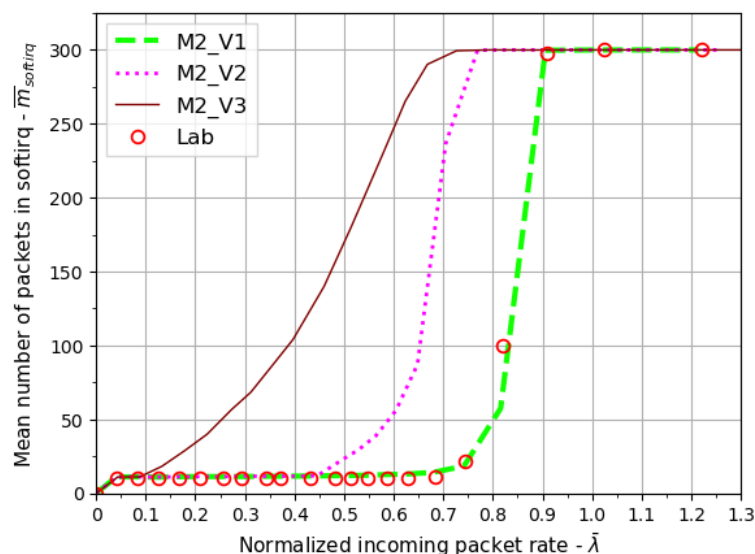


Figure 10. Results of mean number of packets in softirq for different scenarios (V1, V2, and V3).

7. Conclusions

This work proposes an analytical modeling based on a queueing system with vacations to analyze the performance of a Linux packet-capturing system. The concept of vacation allows us to model the behavior of the processor responsible for capturing packets as well as performing additional tasks in the so-called vacations.

The first proposed model, M1, can be considered a simple model that prioritizes the capture process with the exhaustive service discipline ($B \rightarrow \infty$). If the system is not saturated, the execution of additional tasks in the so-called vacations is guaranteed; however, under saturation conditions, the capture process hogs the processor's time and an acceptable capture throughput is achieved to the detriment of performing additional tasks.

The second and third models, M2 and M3, are more complex, and their limited service discipline includes some particularities of the Linux packet capture system (hardirq processing, softirq execution, and budget). The limit of the budget in the softirq allows us to ensure the execution of other tasks, although it causes a loss in terms of capture throughput. In these cases, it will be interesting to assess the tradeoff between having vacations for these additional tasks and the potential loss of capture throughput.

The conclusions of this work are satisfactory with regard to the behavior of the models. The main performance parameter of the capturing system is the throughput. The comparison of throughput results of the limited-service discipline models with the measurements taken from the real probe are positive. Further simulations results suggest that M2 and M3 provide similar performance in terms of throughput.

This work, which combines analytical study with experimental measurements, raises several aspects to consider in the near future. Although in this work we do not do it explicitly, from models M1, M2, and M3, it is also possible to obtain latency results. We believe that the fundamentals of the queueing model with vacations applied in this work could be adapted to environments of VNFs. In 5G network environments, there are multiple software components like VNFs that are connected to provide service chains. We consider that the estimation of the latency of a set of VNFs could be modeled with a queueing system with vacations and we are planning to do this in future work.

Author Contributions: Conceptualization, L.Z., J.D. and A.F.; methodology, L.Z., J.D. and A.F.; software, L.Z. and A.F.; validation, L.Z. and A.F.; formal analysis, L.Z., J.D. and A.F.; investigation, L.Z., J.D. and A.F.; resources, L.Z., J.D. and A.F.; data curation, L.Z., J.D. and A.F.; writing—original draft preparation, L.Z., J.D. and A.F.; writing—review and editing, L.Z., J.D. and A.F.; visualization, L.Z., J.D. and A.F.; supervision, L.Z., J.D. and A.F.; project administration, A.F.; funding acquisition, A.F. All authors have read and agreed to the published version of the manuscript.

Funding: This research was partially supported by the Department of Education of the Basque Government, Spain through the Consolidated Research Groups NQaS (IT1635-22) and MATHMODE (IT1456-22), by the Marie Skłodowska-Curie, Spain grant agreement No 777778, by Grant PID2020-117876RB-I00 funded by MCIN/AEI (10.13039/501100011033) and by Grant KK-2022/00119 funded by the Basque Government.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Stevens, N.T.; Wilson, J.D. The past, present, and future of network monitoring: A panel discussion. *Qual. Eng.* **2021**, *33*, 715–718. [[CrossRef](#)]
2. Li, B.; Springer, J.; Bebis, G.; Gunes, M.H. A survey of network flow applications. *J. Netw. Comput. Appl.* **2013**, *36*, 567–581. [[CrossRef](#)]
3. ntop. Available online: <https://www.ntop.org> (accessed on 24 February 2023).

4. Han, S.; Jang, K.; Park, K.; Moon, S. PacketShader: A GPU-accelerated software router. *ACM SIGCOMM Comput. Commun. Rev.* **2011**, *41*, 195–206.
5. Rizzo, L. netmap: A Novel Framework for Fast Packet I/O. In Proceedings of the USENIX Annual Technical Conference, Boston, MA, USA, 12–15 June 2012; pp. 101–112.
6. Bonelli, N.; Di Pietro, A.; Giordano, S.; Procissi, G. On multi-gigabit packet capturing with multi-core commodity hardware. In *Passive and Active Measurement, Proceedings of the 13th International Conference, PAM 2012, Vienna, Austria, 12–14 March 2012*; Springer: Berlin/Heidelberg, Germany, 2012; pp. 64–73.
7. DPDK Project. Available online: <https://www.dpdk.org> (accessed on 24 February 2023).
8. OpenOnload. Available online: <https://www.openonload.org> (accessed on 24 February 2023).
9. Moreno, V.; Del Rio, P.M.S.; Ramos, J.; Garcia-Dorado, J.L.; Gonzalez, I.; Arribas, F.J.G.; Aracil, J. Packet storage at multi-gigabit rates using off-the-shelf systems. In *Proceedings of the 2014 IEEE International Conference on High Performance Computing and Communications, 2014 IEEE 6th International Symposium on Cyberspace Safety and Security, 2014 IEEE 11th International Conference on Embedded Software and Syst (HPCC, CSS, ICSS), Paris, France, 20–22 August 2014*; IEEE: Piscataway, NJ, USA, 2014; pp. 486–489.
10. Zabala, L.; Doncel, J.; Ferro, A. Optimality of a Network Monitoring Agent and Validation in a Real Probe. *Mathematics* **2023**, *11*, 610. [[CrossRef](#)]
11. Giambene, G. *Queueing Theory and Telecommunications: Networks and Applications*; Springer Nature: Berlin/Heidelberg, Germany, 2021.
12. Poryazov, S.; Saranova, E.; Ganchev, I. Conceptual and analytical models for predicting the quality of service of overall telecommunication systems. In *Autonomous Control for a Reliable Internet of Services: Methods, Models, Approaches, Techniques, Algorithms, and Tools*; Springer International Publishing: Cham, Switzerland, 2018; pp. 151–181.
13. Andonov, V.; Poryazov, S.; Otsetova, A.; Saranova, E. A queue in overall telecommunication system with quality of service guarantees. In Proceedings of the Future Access Enablers for Ubiquitous and Intelligent Infrastructures: 4th EAI International Conference, FABULOUS 2019, Sofia, Bulgaria, 28–29 March 2019; Proceedings 283; Springer: Berlin/Heidelberg, Germany, 2019; pp. 243–262.
14. Rotter, C.; Van Do, T. A queueing model for threshold-based scaling of UPF instances in 5G core. *IEEE Access* **2021**, *9*, 81443–81453. [[CrossRef](#)]
15. Hsieh, C.Y.; Phung-Duc, T.; Ren, Y.; Chen, J.C. Design and analysis of dynamic block-setup reservation algorithm for 5G network slicing. *IEEE Trans. Mob. Comput.* **2022**, *1*. [[CrossRef](#)]
16. Agarwal, S.; Malandrino, F.; Chiasserini, C.F.; De, S. VNF placement and resource allocation for the support of vertical services in 5G networks. *IEEE/ACM Trans. Netw.* **2019**, *27*, 433–446. [[CrossRef](#)]
17. Keramidi, I.; Uzunidis, D.; Moscholios, I.; Sarigiannidis, P.; Logothetis, M. On Queueing Models for the Performance Analysis of a Vehicular Ad Hoc Network. In Proceedings of the 2022 International Conference on Software, Telecommunications and Computer Networks (SoftCOM), Split, Croatia, 22–24 September 2022; IEEE: Piscataway, NJ, USA, 2022; pp. 1–6.
18. Kartashevskiy, V.; Buranova, M. OpenFlow-based software-defined networking queue model. In Proceedings of the Distributed Computer and Communication Networks: 24th International Conference, DCCN 2021, Moscow, Russia, 20–24 September 2021; Revised Selected Papers; Springer: Berlin/Heidelberg, Germany, 2022; pp. 62–76.
19. Aliyu, A.L.; Diocou, J. An Analytical Queueing Model Based on SDN for IoT Traffic in 5G. In *Advanced Information Networking and Applications, Proceedings of the 37th International Conference on Advanced Information Networking and Applications (AINA-2023), Juiz de Fora, Brazil, 29–31 March 2023*; Springer: Berlin/Heidelberg, Germany, 2023; Volume 3, pp. 435–445.
20. Wu, W.; Crawford, M.; Bowden, M. The performance analysis of Linux networking—Packet receiving. *Comput. Commun.* **2007**, *30*, 1044–1057. [[CrossRef](#)]
21. Salah, K.; Elbadawi, K.; Boutaba, R. Performance modeling and analysis of network firewalls. *Netw. Serv. Manag. IEEE Trans.* **2012**, *9*, 12–21. [[CrossRef](#)]
22. Zapechnikov, S.; Miloslavskaya, N.; Tolstoy, A. Modeling of next-generation firewalls as queueing services. In Proceedings of the 8th International Conference on Security of Information and Networks, Vienna, Austria, 23–24 May 2015; pp. 250–257.
23. Li, X.; Ren, F.; Yang, B. Modeling and analyzing the performance of high-speed packet I/O. *Tsinghua Sci. Technol.* **2021**, *26*, 426–439. [[CrossRef](#)]
24. Tian, N.; Zhang, Z.G. *Vacation Queueing Models: Theory and Applications*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2006; Volume 93.
25. Takagi, H. *Queueing Analysis. A Foundation of Performance Evaluation Volume 1: Vacation and Priority Systems (Part 1)*; Elsevier Science Publishers B.V.: Amsterdam, The Netherlands, 1999.
26. Lee, T.T. M/G/1/N queue with vacation time and exhaustive service discipline. *Oper. Res.* **1984**, *32*, 774–784. [[CrossRef](#)]
27. Lee, T.T. M/G/1/N queue with vacation time and limited service discipline. *Perform. Eval.* **1989**, *9*, 181–190. [[CrossRef](#)]
28. Benvenuti, C. *Understanding Linux Network Internals*; O'Reilly Media, Inc.: Sebastopol, CA, USA, 2006.
29. The Linux Kernel Documentation. Available online: <https://docs.kernel.org> (accessed on 24 February 2023).
30. The Linux Documentation Project. Available online: <https://tldp.org> (accessed on 24 February 2023).
31. Bolch, G.; Greiner, S.; De Meer, H.; Trivedi, K.S. *Queueing Networks and Markov Chains: Modeling and Performance Evaluation with Computer Science Applications*; John Wiley & Sons: Hoboken, NJ, USA, 2006.
32. Adan, I.; Resing, J. *Queueing Theory*; University of Twente: Enschede, The Netherlands, 2002.
33. Stewart, W.J. *Probability, Markov Chains, Queues, and Simulation: The Mathematical Basis of Performance Modeling*; Princeton University Press: Princeton, NJ, USA, 2009.

34. Cooper, R.B. Queueing theory. In Proceedings of the ACM'81 Conference, Los Angeles, CA, USA, 9–11 November 1981; pp. 119–122.
35. Munoz, A.; Ferro, A.; Liberal, F.; Lopez, J. A Kernel-Level Monitor over Multiprocessor Architectures for High-Performance Network Analysis with Commodity Hardware. In Proceedings of the 2007 International Conference on Sensor Technologies and Applications (SENSORCOMM 2007), Valencia, Spain, 14–20 October 2007; IEEE: Piscataway, NJ, USA, 2007; pp. 457–462.
36. Pineda, A.; Zabala, L.; Ferro, A. Network architecture to automatically test traffic monitoring systems. In Proceedings of the Mosharaka International Conference on Communications and Signal Processing (MIC-CSP2012), Barcelona, Spain, 6–8 April 2012.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.