JITEL 2017
XIII Jornadas de Ingeniería Telemática
27-29 Septiembre 2017, Valencia (España)

Actas de las XIII Jornadas de Ingeniería Telemática (JITEL 2017), Valencia (España), 27-29 de Septiembre de 2017.

ISBN: 978-84-9048-595-8
DOI: http://dx.doi.org/10.4995/JITEL2017.2017.6580

Modelo de colas con *vacations* aplicado a un sistema de captura de paquetes

Luis Zabala, Armando Ferro, Ander Nieva.
Departamento de Ingeniería de Comunicaciones.
Universidad del País Vasco/Euskal Herriko Unibertsitatea (UPV/EHU).
ESI Bilbao. Alameda de Urquijo s/n. 48013 Bilbao.
luis.zabala@ehu.eus, armando.ferro@ehu.eus, ander.nieva@ehu.eus.

***Resumen*—La mejora de sistemas de captura de paquetes de red ha sido extensamente cubierta como tema de investigación en los pasados años. La mayoría de estas iniciativas han sido respaldadas por evaluaciones experimentales; sin embargo, ha habido pocas propuestas de modelado. Este trabajo presenta el modelado y el análisis de un sistema de cola finito con *vacations* aplicado a la etapa de captura de paquetes de un sistema de monitorización de red. Se plantean dos modelos con disciplina de servicio diferente (exhaustiva y limitada) y se evalúan sus rendimientos, principalmente en forma de throughput de captura, para distintos escenarios. Éstos contemplan diferentes tasas de entrada de paquetes y tiempos de *vacation*. Los resultados teóricos, derivados de un estudio analítico basado en ecuaciones de balance y su desarrollo en forma matricial, también son comparados con los de una sonda real de tráfico de red que captura paquetes.**

***Palabras Clave*—motor de captura, vacation, teoría de colas**

I. INTRODUCCIÓN

Un sistema finito de colas con *vacations* puede ser útil para modelar y analizar el rendimiento de un sistema de captura de tráfico situado en un entorno de monitorización de red. En un modelo de colas clásico, los servidores siempre están disponibles. Sin embargo, puede haber sistemas de colas reales donde los servidores pueden dejar de estar disponibles durante un cierto periodo de tiempo debido a razones varias. Para analizar estos sistemas, se introduce el estado de *vacation* [1], que representa el periodo de la ausencia temporal del servidor. Por tanto, en este tipo de sistemas, después de cada periodo activo en el que el servidor atiende a los elementos de la cola principal, el servidor pasa a ejecutar tareas adicionales que no están relacionadas con los clientes de la cola principal [2].

En todo sistema de monitorización de red se tiene una primera etapa de captura de paquetes donde se recoge de la red, "en bruto", los datos de medida [3]. Posteriormente esos datos pueden ser analizados en detalle y procesados para extraer de ellos interpretaciones de nivel superior. Finalmente, los resultados extraídos del análisis son presentados en diferentes formatos a los operadores de red. Las compañías de telecomunicaciones invierten grandes cantidades de dinero en monitorización de tráfico con el objetivo de garantizar la satisfacción de sus clientes y, al mismo tiempo, hacer crecer su cuota de mercado [4].

El objeto de estudio de este trabajo es el modelado de la etapa de captura de paquetes, teniendo presente que el sistema puede tener funciones adicionales de monitorización que realizar. Por ello, se propone un modelado de colas con *vacation*, donde la tarea de captura se representa mediante el servicio de la cola principal y las otras funciones que podría llevar a cabo el sistema son ejecutadas por el mismo procesador durante sus tiempos de *vacation*. Estos modelos ayudan a estimar el rendimiento de la etapa de captura y ver cómo influye sobre él los tiempos de *vacation*, es decir, la dedicación a otras tareas. Este modelado cobra mayor interés cuando el sistema se encuentra en condiciones de saturación, por ejemplo, cuando las tasas de transmisión de la red crecen.

En el ámbito de la captura de paquetes, el incremento de las tasas de transmisión de datos es continuo. Esto hace que la implementación de sistemas de monitorización capaces de afrontar este ritmo creciente de las redes sea una ardua labor, incluso para el caso en el que las aplicaciones que se ejecutan en la capa superior del sistema de monitorización llevan a cabo un procesamiento mínimo. La monitorización de tráfico en el rango entre 100 Mbps y 1 Gbps fue considerada todo un reto hace muy pocos años, mientras que los routers comerciales actuales ya tienen interfaces de 10, 40 ó incluso 100 Gbps.

Por otro lado, es un hecho que, en los últimos años, se ha producido una mejora en el rendimiento de los sistemas que procesan tráfico de red implementados sobre hardware de propósito general. Esta mejora se debe tanto a las mejoras de software como a la evolución de hardware (aumento de velocidades de las tarjetas de red, aumento de la frecuencia de reloj de la CPU, aparición de CPUs multinúcleo, nuevas características de las tarjetas de red que hacen posible ahorrar ciclos de CPU, etc.) [5].

Así, ha habido varias propuestas de motores de captura basados en este tipo de arquitecturas como PF_RING [6], PacketShader [7], Netmap [8], PFQ [9], OpenOnLoad [10] y HPCAP [11]. Todo esto hace que los sistemas basados en hardware genérico sean muy atractivos para la monitorización de tráfico de redes de alta velocidad, puesto que su rendimiento puede ser comparado con el de hardware especializado (FPGAs, *network processors*, o soluciones comerciales proporcionadas por fabricantes de routers) y, además, su precio es significativamente menor.

Sin embargo, existen pocas propuestas de modelado para estas soluciones de captura. Entre ellas, se pueden destacar algunas relacionadas con la caracterización de sistemas basados en Linux [12] y otras con sistemas de captura y análisis de tráfico [13].

En definitiva, la existencia de numerosas propuestas de motores de captura, pero la escasez de propuestas de modelado y de análisis del caso en el que otras tareas pueden influir en el rendimiento del sistema de captura, motivan este trabajo donde se pretende aplicar un modelo de cola finita con *vacations* a un sistema de captura de paquetes. Además, tal y como se verá, también se propone un modelo para el caso de un sistema Linux, debido a su carácter abierto y por ser la base de muchos motores de captura sobre hardware de propósito general. Parece de interés construir modelos matemáticos que permitan estudiar teóricamente el rendimiento de sistemas de captura de tráfico, con objeto de contribuir en futuras mejoras de diseño destinadas a este tipo de sistemas.

El resto del artículo está estructurado de la siguiente forma. La sección II presenta un modelo para un sistema de captura genérico con disciplina de servicio exhaustiva, mientras que el modelo expuesto en la sección III es más específico ya que recoge las características de un sistema de captura de paquetes basado en Linux. La sección IV proporciona resultados de los modelos. Finalmente, la sección V expone las conclusiones.

II. MODELO CON *VACATIONS* Y DISCIPLINA DE SERVICIO EXHAUSTIVA

El primer modelo que se presenta, M1, se basa en un modelo de cola finita con una serie de hipótesis markovianas. La llegada de paquetes al sistema de captura sigue un proceso de Poisson con tasa λ. Estos paquetes son los que provienen de la tarjeta de red y son transferidos, vía DMA (Direct Memory Access), a un área de memoria de tamaño finito. A continuación, el tratamiento de paquetes por parte del motor de captura se representa mediante un único servidor que se dedica a esta tarea en su periodo activo y a otras en su periodo de *vacation*. El tiempo dedicado a capturar paquetes sigue una distribución exponencial de media $1/\mu_S$, mientras que los tiempos de *vacation* una distribución exponencial de media $1/\mu_V$. Se denominará N al número máximo de paquetes que puede haber en la etapa de captura. Esto tiene en cuenta el paquete que está siendo atendido por el procesador más los que están en la cola de espera. Si hay N paquetes en el sistema de captura, los nuevos paquetes entrantes serán bloqueados, es decir, dichos paquetes no podrán introducirse en el búfer, ya que éste está completo, y serán rechazados. Por último, el proceso de captura no terminará hasta que se vacíe el búfer, momento en el que comenzará un periodo de *vacation*. Por tanto, se considera una disciplina de servicio exhaustiva.

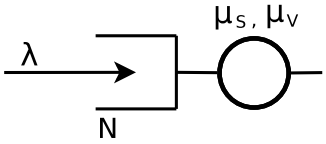

Fig. 1. Modelo de cola finita con *vacations*

A. Cadena de Markov asociada a la etapa de captura

Este primer modelo de cola finita con *vacations* se basa en una cadena de Markov con un espacio de estados $S = \{(n,m), 0 \leq n \leq N, 0 \leq m \leq 1\}$. La variable n indica el número de paquetes que hay en la etapa de captura; m identifica si el servidor está en un periodo activo capturando paquetes (en cuyo caso, $m = 1$), o si está en un periodo de *vacation* ($m = 0$).

La Fig. 2 muestra el diagrama de transiciones de estado en función de las tasas λ, μ_S y μ_V. Primeramente, hay que indicar que si el sistema está vacío, estado $(0,0)$, y llega un paquete nuevo, debido a la política del planificador de tareas, el inicio del proceso de captura no es inmediato, sino que se produce después de una *vacation*. Por este motivo, se da la transición de $(0,0)$ a $(1,0)$ con tasa λ. Durante la *vacation* el proceso está en los estados $(n,0)$, donde el número de paquetes aumenta con tasa λ, salvo en $(N,0)$ por haber bloqueo, y también es posible terminar la *vacation* con tasa μ_V cuando el planificador lo decida.

En los estados $(n,1)$, el procesador está capturando paquetes. Entonces, pueden llegar nuevos paquetes con tasa λ, salvo en $(N,1)$, y salir paquetes del sistema de captura con tasa μ_S. La finalización del proceso de captura sólo se da cuando se vacía el búfer: transición de $(1,1)$ a $(0,0)$ en la Fig. 2.

B. Ecuaciones de balance y probabilidades de estado

Sea $\pi_{n,m}$, la probabilidad del estado (n,m) en régimen estacionario. Se plantean las siguientes ecuaciones de balance [14] asociadas a los estados de la Fig. 2.

$$\lambda\pi_{0,0} = \mu_S\pi_{1,1} \tag{1a}$$

$$(\lambda+\mu_V)\pi_{n,0} = \lambda\pi_{n-1,0}, \quad 1 \leq n \leq N-1 \tag{1b}$$

$$(\lambda+\mu_S)\pi_{n,1} = \lambda\pi_{n-1,1} + \mu_V\pi_{n,0} + \mu_S\pi_{n+1,1}, \quad 1 \leq n \leq N-1 \tag{1c}$$

$$\mu_V\pi_{N,0} = \lambda\pi_{N-1,0} \tag{1d}$$

$$\mu_S\pi_{N,1} = \mu_V\pi_{N,0} + \lambda\pi_{N-1,1} \tag{1e}$$

Por otro lado, utilizando el principio de balance global [15], por el que el flujo saliente de un conjunto de estados es igual al flujo entrante a dicho conjunto

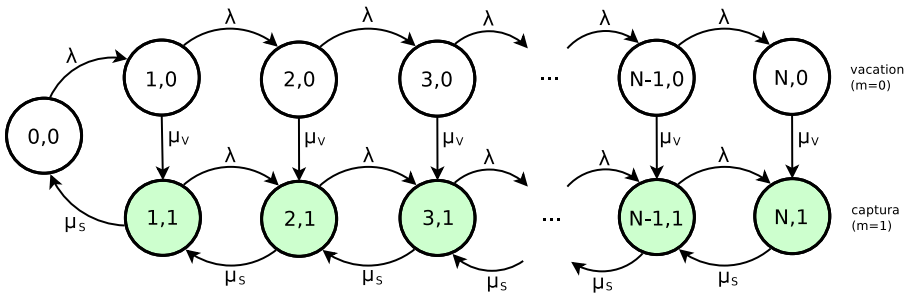

Fig. 2. Cadena de Markov del modelo M1

$$\lambda(\pi_{n,0}+\pi_{n,1})=\mu_S\pi_{n+1,1},\quad 0\le n\le N-1 \tag{2}$$

Y dado que la suma de probabilidades debe ser 1.

$$\sum_{n=0}^{N}(\pi_{n,0}+\pi_{n,1})=1 \tag{3}$$

Tras agrupar y reescribir las ecuaciones (1a)-(1e) y (2), se llega a una solución geométrica matricial

$$\begin{pmatrix}\pi_{n,0}\\ \pi_{n,1}\end{pmatrix}=R^n\begin{pmatrix}\pi_{0,0}\\ 0\end{pmatrix}\quad 1\le n\le N \tag{4}$$

$$\text{con}\quad R=\begin{pmatrix}\lambda/(\lambda+\mu_V) & 0\\ \lambda/\mu_S & \lambda/\mu_S\end{pmatrix}$$

Imponiendo la condición de normalización (3), se obtiene el valor de $\pi_{0,0}$.

$$\pi_{0,0}=\frac{1}{S};\quad S=\begin{pmatrix}1 & 1\end{pmatrix}\left(\sum_{n=0}^{N}R^N\right)\begin{pmatrix}1\\ 0\end{pmatrix} \tag{5}$$

$$\text{siendo}\quad \sum_{n=0}^{N}R^N=(I-R)^{-1}\left(I-R^{N+1}\right)$$

Después de calcular $\pi_{0,0}$ ya se pueden determinar todas las probabilidades de estado $\pi_{n,m}$ aplicando (4).

C. Parámetros de rendimiento

Una vez conocidas las probabilidades de estado en régimen estacionario, $\pi_{n,m}$, ya es posible calcular los parámetros de rendimiento de la etapa de captura: probabilidad de bloqueo, throughput, utilización de CPU en captura, disponibilidad de CPU, etc.

Probabilidad de bloqueo (P_B). Es la probabilidad de perder paquetes cuando el búfer está lleno. Esto sucede en los estados $(N,0)$ y $(N,1)$.

$$P_B=\pi_{N,0}+\pi_{N,1} \tag{6}$$

Throughput de captura, (X_C). Se define como el número medio de paquetes capturados por segundo. La captura tiene lugar en los estados $(n,1)$.

$$X_C=\mu_S\sum_{n=1}^{N}\pi_{n,1}=\lambda\left(1-P_B\right) \tag{7}$$

Frecuencia de *captura* (f_C) Mide el número de procesos de captura que se ejecutan por unidad de tiempo. La frecuencia f_C está relacionada con el ciclo compuesto por un periodo activo de captura y una *vacation*. Teniendo en cuenta el carácter poissoniano de las llegadas de paquetes, llamando T_C a la duración media de un proceso de captura y T_{ciclo} al tiempo medio del ciclo

$$T_{ciclo}=\frac{1}{\lambda}+\frac{1}{\mu_V}+T_C \tag{8}$$

Y la frecuencia de captura queda

$$T_{ciclo}\cdot\pi_{0,0}=\frac{1}{\lambda}\Rightarrow f_C=\frac{1}{T_{ciclo}}=\lambda\pi_{0,0} \tag{9}$$

Tiempo medio de captura (T_C). Es la duración media de un periodo activo del procesador responsable de la etapa de captura. Su expresión se obtiene de la siguiente manera.

$$T_C=T_{ciclo}\sum_{n=1}^{N}\pi_{n,1}=\frac{\sum_{n=1}^{N}\pi_{n,1}}{f_C} \tag{10}$$

III. MODELO CON *VACATIONS* Y DISCIPLINA DE SERVICIO LIMITADA

El segundo modelo propuesto, M2, tiene como objeto representar un sistema más específico y se ha tomado como ejemplo el sistema de captura de paquetes de Linux. Aquí, el proceso que lleva a cabo esta tarea se denomina *softirq* y tiene un límite denominado *budget* que establece el número máximo de paquetes que pueden ser tratados en una *softirq*. Por ello, si se alcanza el *budget*, el procesador deja la *softirq* y pasa a ejecutar otra tarea [18]. Para representar este comportamiento, se propone el modelo M2 basado en una disciplina de servicio limitada, en contraposición a la exhaustiva del modelo M1. El resto de suposiciones coinciden con las del modelo M1. Por tanto, se mantienen las tasas λ, μ_S y μ_V, así como el tamaño de buffer N y se añade el parámetro B, que denota al valor del *budget*.

Un segundo elemento diferencial de M2 es que se tienen en consideración los tiempos consumidos en en rutinas de atención a *hardirq* de Linux [18]. Se supone que el tiempo de ejecución de una rutina asociada a *hardirq* sigue una distribución exponencial con media $1/\mu_H$.

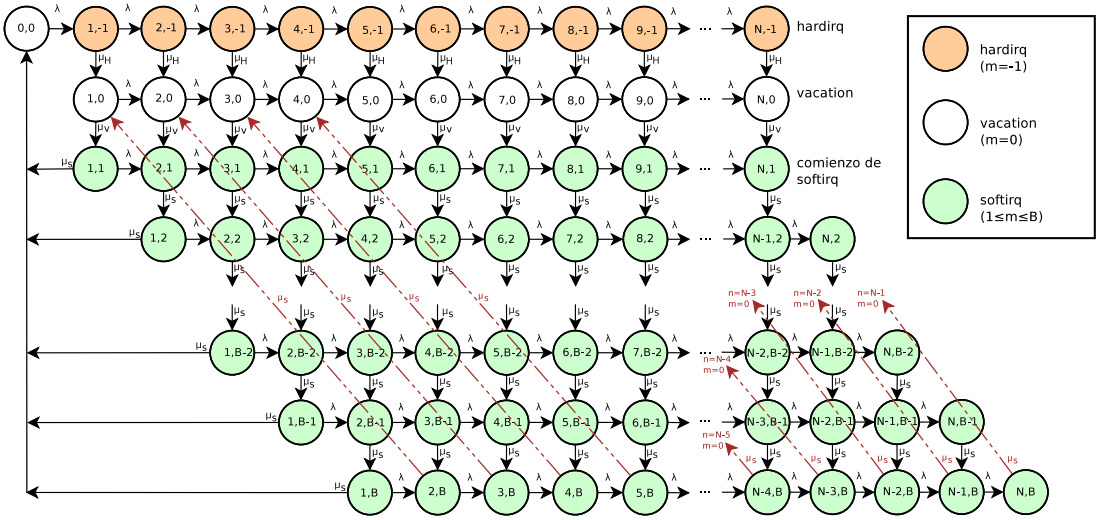

Fig. 3. Diagrama de estados del modelo M2

A. Cadena de Markov asociada a la etapa de captura

M2, modelo de cola finita con *vacations* y disciplina de servicio limitada, se fundamenta en una cadena de Markov con $S = \{(n,m), 0 \leq n \leq N, -1 \leq m \leq B\}$, donde n indica el número de paquetes de la etapa de captura y m identifica diferentes estados del procesador. Así, $m = -1$ indica que el procesador está ejecutando la rutina de servicio de una *hardirq*; $m = 0$ indica que está en *vacation*; y si $m \in [1, B]$, ello indica que el procesador está ejecutando una *softirq* y, concretamente, que se está capturando el paquete número *m* en dicha *softirq*. El valor máximo de m es el valor del *budget*, B.

El diagrama de estados y transiciones de la Fig. 3 refleja el procedimiento del sistema de captura de Linux. Primero, si el sistema está vacío, estado $(0,0)$, y llega un paquete, según el procedimiento de Linux, la llegada de este primer paquete provoca la ejecución de la rutina de atención de *hardirq*. Por ello, en la Fig. 3 se tiene la transición de $(0,0)$ a $(1,-1)$ con tasa λ. La rutina de *hardirq* suele ser corta y, en ella, se deshabilitan las *hardirq* de red, para evitar que llegadas posteriores de paquetes vuelvan a interrumpir al procesador, y se planifica la ejecución de una *softirq*, que tendrá lugar más adelante. Durante la *hardirq*, el sistema está en los estados $(n,-1)$. Pueden llegar nuevos paquetes con tasa λ, exceptuando en $(N,-1)$ o puede finalizarse la propia *hardirq* con tasa μ_H. Esto último implica una transición hacia un estado de *vacation* $(n,0)$. El comportamiento de las *vacation* de M2 es idéntico a las de M1, por lo que, cuando el planificador de tareas lo determina, acaba la *vacation* con tasa μ_V y comienza la ejecución de una *softirq*. Esto último conlleva la transición de un estado $(n,0)$a un estado $(n,1)$.

La ejecución de una *softirq* está representada por el conjunto de estados (n,m) donde $1 \leq n \leq N$ y $1 \leq m \leq B$. Durante la *softirq*, pueden llegar nuevos paquetes con tasa λ y, por tanto, producir transiciones de (n,m) a $(n+1,m)$, salvo en los estados de bloqueo (N,m). También es posible la salida de paquetes procesados, lo que conlleva transiciones de (n,m) a $(n-1,m+1)$. Otro aspecto importante a considerar es la finalización de la *softirq* para la que hay dos opciones. Una es que la cola se vacía y, según el procedimiento de Linux, se habilitan de nuevo las *hardirq*. En el modelo, este caso corresponde a la transición de un estado $(1,m)$ al estado $(0,0)$. La segunda opción es que se alcanza el *budget*, estado (n,B), caso en el que Linux planifica una nueva *softirq* sin deshabilitar las *hardirq* y, en el modelo, se pasa de un estado (n,B) a un estado $(n-1,0)$, es decir, a un estado de *vacation*.

B. Ecuaciones de balance y probabilidades de estado

En régimen estacionario, las probabilidades $\pi_{n,m}$ satisfacen las ecuaciones de balance de los estados de la Fig. 3. En primer lugar, en el estado $(0,0)$,

$$\lambda \pi_{0,0} = \mu_S \sum_{m=1}^{B} \pi_{1,m} \tag{11}$$

En los estados $(1,m)$,

$$(\lambda + \mu_H)\,\pi_{1,-1} = \lambda \pi_{0,0} \tag{12a}$$
$$(\lambda + \mu_V)\,\pi_{1,0} = \mu_H \pi_{1,-1} + \mu_S \pi_{2,B} \tag{12b}$$
$$(\lambda + \mu_S)\,\pi_{1,1} = \mu_V \pi_{1,0} \tag{12c}$$
$$(\lambda + \mu_S)\,\pi_{1,m} = \mu_S \pi_{2,m-1} \tag{12d}$$
$$2 \leq m \leq B-1$$
$$(\lambda + \mu_S)\,\pi_{1,B} = \mu_S \pi_{2,B-1} \tag{12e}$$

Para los estados (n,m) con $2 \leq n \leq N-1$,

$$(\lambda+\mu_H)\,\pi_{n,-1} = \lambda\pi_{n-1,-1} \tag{13a}$$
$$(\lambda+\mu_V)\,\pi_{n,0} = \lambda\pi_{n-1,0} + \mu_H\pi_{n,-1} + \mu_S\pi_{n+1,B} \tag{13b}$$
$$(\lambda+\mu_S)\,\pi_{n,1} = \lambda\pi_{n-1,1} + \mu_V\pi_{n,0} \tag{13c}$$
$$(\lambda+\mu_S)\,\pi_{n,m} = \lambda\pi_{n-1,m} + \mu_S\pi_{n+1,m-1} \quad 2 \le m \le (B-1) \tag{13d}$$
$$(\lambda+\mu_S)\,\pi_{n,B} = \lambda\pi_{n-1,B} + \mu_S\pi_{n+1,B-1} \tag{13e}$$

Y en los estados (N, m),

$$\mu_H\pi_{N,-1} = \lambda\pi_{N-1,-1} \tag{14a}$$
$$\mu_V\pi_{N,0} = \lambda\pi_{N-1,0} + \mu_H\pi_{N,-1} \tag{14b}$$
$$\mu_S\pi_{N,1} = \lambda\pi_{N-1,1} + \mu_V\pi_{N,0} \tag{14c}$$
$$\mu_S\pi_{N,m} = \lambda\pi_{N-1,m} \quad 2 \le m \le (B-1) \tag{14d}$$
$$\mu_S\pi_{N,B} = \lambda\pi_{N-1,B} \tag{14e}$$

Además, se tiene la condición de normalización.

$$\pi_{0,0} + \sum_{n=1}^{N}\sum_{m=-1}^{B}\pi_{n,m} = 1 \tag{15}$$

Para calcular las probabilidades $\pi_{n,m}$, se realiza un desarrollo matricial para el que se definen los siguientes vectores de probabilidades de dimensión $(B+2)\times 1$

$$\vec{\pi}_0 = \begin{pmatrix} \pi_{0,0} \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{pmatrix} \quad \vec{\pi}_n = \begin{pmatrix} \pi_{n,-1} \\ \pi_{n,0} \\ \pi_{n,1} \\ \vdots \\ \pi_{n,B-1} \\ \pi_{n,B} \end{pmatrix} \quad 1 \le n \le N \tag{16}$$

El desarrollo concluye con unas relaciones matriciales entre los vectores de probabilidades de forma que todos ellos quedan en función de la probabilidad $\pi_{0,0}$.

$$\begin{cases} \vec{\pi}_1 = Z_1\vec{\pi}_0 \\ \vec{\pi}_n = Z_n\vec{\pi}_{n-1} \quad 2 \le n \le N \end{cases} \tag{17}$$

A su vez, las matrices Z_n son resultado de operaciones matriciales donde intervienen las tasas λ, μ_S, μ_V, y μ_H.

$$\begin{cases} Z_N = -\lambda A^{-1} \\ Z_n = -\lambda\left(A - \lambda I + BZ_{n+1}\right)^{-1} \quad 2 \le n \le N-1 \\ Z_1 = -\left(A - \lambda I + BZ_2\right)^{-1} C \end{cases} \tag{18}$$

Las matrices A, B y C, que aparecen en (18), son cuadradas, de dimensiones $(B+2)\times(B+2)$, y contienen los siguientes elementos:

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

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

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

Sustituyendo $\pi_{n,m}$ $(1 \le n \le N, -1 \le m \le B)$ en la condición de normalización (15) por las expresiones extraídas de (16)-(18), se obtiene el valor de $\pi_{0,0}$.

$$\pi_{0,0} = \frac{1}{S}; \quad S = 1 + \left[\sum_{n=1}^{N}\vec{e}_1^{\,T}\left(\prod_{i=0}^{n-1}Z_{n-i}\right)\vec{e}_2\right] \tag{19}$$

$\vec{e}_1$ y $\vec{e}_2$ son vectores de dimensiones $(B+2)\times 1$.

$$\begin{aligned} \vec{e}_1^{\,T} &= (1 \quad 1 \quad \dots\dots \quad 1) \\ \vec{e}_2^{\,T} &= (1 \quad 0 \quad \dots\dots \quad 0) \end{aligned} \tag{20}$$

Una vez que se tiene el valor de $\pi_{0,0}$, ya se pueden determinar todas las probabilidades de estado $\pi_{n,m}$, aplicando (16).

C. Parámetros de rendimiento

Con las probabilidades de estado en régimen estacionario, $\pi_{n,m}$, ya es posible calcular los parámetros de rendimiento de interés.

Probabilidad de bloqueo (P_B). Es igual a la suma de las probabilidades de los estados de bloqueo (N, m).

$$P_B = \sum_{m=-1}^{B}\pi_{N,m} \tag{21}$$

Throughput de captura (X_C). Tiene en cuenta la probabilidad de estar ejecutando una *softirq* y la tasa de servicio μ_S.

$$X_C = \mu_S \sum_{n=1}^{N} \sum_{m=1}^{B} \pi_{n,m} \tag{22}$$

Utilización de procesador en captura (U_C). Tiene en cuenta los estados en los que el procesador se dedica a *hardirqs* y *softirqs*.

$$U_C = \sum_{n=1}^{N} \pi_{n,-1} + \sum_{n=1}^{N} \sum_{m=1}^{B} \pi_{n,m} \tag{23}$$

Directamente relacionada con la utilización, se tiene la disponibilidad del procesador para otras tareas distintas de la captura, D.

$$D = \pi_{0,0} + \sum_{n=1}^{N} \pi_{n,0} = 1 - U_C \tag{24}$$

Frecuencia de *hardirq* (f_{hirq}). Este parámetro mide el número de veces por unidad de tiempo que se atiende a la rutina de atención a la *hardirq*. Para calcular dicha frecuencia, se estima T_{ciclo}, tiempo medio del ciclo compuesto por sistema vacío, *hardirq* y conjunto de *vacations* y *softirqs* hasta vaciar de nuevo el sistema.

$$T_{ciclo} = \frac{1}{\lambda} + \frac{1}{\mu_H} + k_s \left(\frac{1}{\mu_V} + T_{sirq} \right) \tag{25}$$

T_{sirq} es el tiempo medio de *softirq* y k_s es el número medio de *softirqs* dentro del ciclo. Dado que se ha asumido que las llegadas siguen un proceso de Poisson:

$$\begin{aligned} \frac{1}{\lambda} &= T_{ciclo} \cdot \pi_{0,0} \\ f_{hirq} = \frac{1}{T_{ciclo}} &= \lambda \cdot \pi_{0,0} \end{aligned} \tag{26}$$

Frecuencia de *softirq* (f_{sirq}). Es equivalente a la frecuencia de captura (f_C) definida en el modelo M1. En M2, la expresión que toma es la siguiente:

$$f_{sirq} = \mu_V \sum_{n=1}^{N} \pi_{n,0} \tag{27}$$

Tiempo medio de *softirq* (T_{sirq}). Es la duración media de una *softirq*. Coincide con el periodo activo del procesador en captura.

$$T_{sirq} = \frac{\sum_{n=1}^{N} \sum_{m=1}^{B} \pi_{n,m}}{\mu_V \sum_{n=1}^{N} \pi_{n,0}} \tag{28}$$

Una vez calculado el tiempo medio de *softirq*, se puede estimar el número medio de paquetes tratados en una *softirq*, $\overline{m}_{sirq}$.

$$\overline{m}_{sirq} = \frac{T_{sirq}}{\frac{1}{\mu_S}} = \mu_S T_{sirq} = \frac{\mu_S \sum_{n=1}^{N} \sum_{m=1}^{B} \pi_{n,m}}{\mu_V \sum_{n=1}^{N} \pi_{n,0}} \tag{29}$$

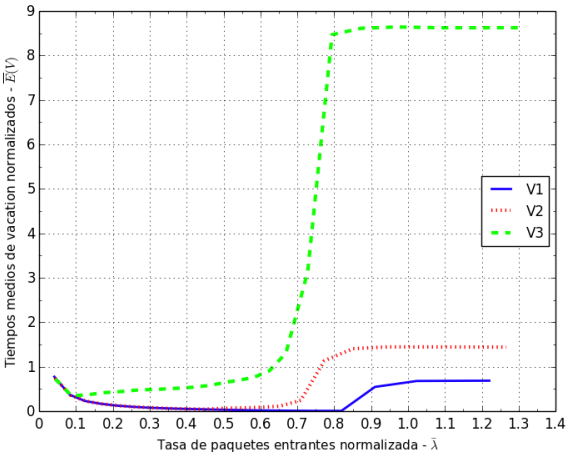

Fig. 4. Escenarios de evaluación V1, V2 y V3

IV. EVALUACIÓN DE MODELOS

En este apartado se muestran resultados de la evaluación de los modelos M1 y M2. Las curvas analíticas se han obtenido a través de la implementación en MATLAB de las ecuaciones derivadas de los modelos. Asimismo, con objeto de realizar comparativas, se presentan valores experimentales obtenidos con una sonda real de captura y análisis de paquetes basada en Linux [16] que opera dentro de una plataforma de medidas de laboratorio [17]. Para que sean comparables ambos tipos de resultados, algunos parámetros de entrada de los modelos (λ, $1/\mu_S$, $1/\mu_V$, $1/\mu_H$) toman sus valores a partir de mediciones realizadas sobre la plataforma experimental.

Se definen tres escenarios de evaluación (V1, V2 y V3) que se caracterizan por tener distinto comportamiento de *vacation* tal y como se muestra en la Fig. 4. Concretamente, se representa, para diferentes tasas normalizadas de entrada de paquetes, $\overline{\lambda} = \lambda/\mu_S$, el tiempo medio de *vacation* normalizado con respecto a la duración máxima de una *softirq* de Linux, $\overline{E}(V) = (1/\mu_V)/(B/\mu_S)$, con $B = 300$ ya que éste es el valor típico de *budget*. En la Fig. 4 se aprecia que los tiempos $\overline{E}(V)$ no permanecen constantes. Esto se debe a que son tiempos extraídos de la sonda real y, en ésta, las operaciones de análisis posteriores a la captura requieren mayores consumos computacionales a medida que aumenta la tasa de entrada de paquetes. Únicamente se ha reproducido el escenario V1 en laboratorio, por lo que, solamente para este caso se realizan comparativas entre los modelos (M1 y M2) y los valores de la sonda.

La Fig. 5 muestra resultados de medidas de rendimiento, tanto de la evaluación de los modelos M1 y M2 como de las medidas de la sonda real de laboratorio (referidas como "Lab"). En primer lugar, la Fig. 5a expone cómo varía el throughput de captura normalizado, $\overline{X}_C = X_C/\mu_S$, con los diferentes escenarios. Se evalúa con $N = 200$ y $B = 300$ (valores coincidentes con la configuración del driver de la tarjeta de red de la sonda). Se observa que el modelo M1 tiene un comportamiento similar para los

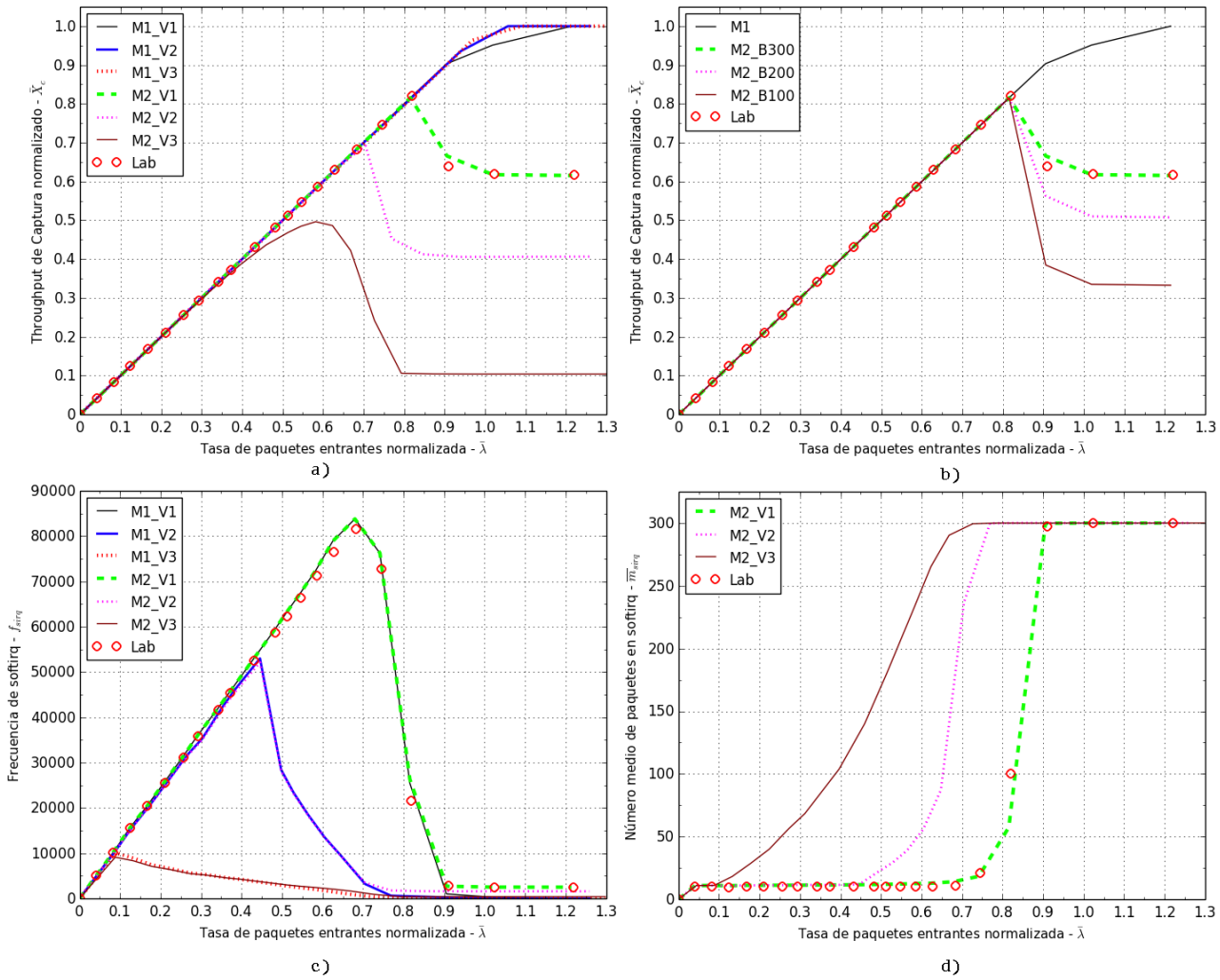

Fig. 5. Medidas de rendimiento de los modelos M1 y M2

escenarios V1, V2, V3; es decir, prácticamente no se ve afectado por los tiempos de *vacation*. Esto se debe a la disciplina exhaustiva que se ha supuesto para el proceso de captura, que permite alargar indefinidamente el tiempo dedicado a la captura y obtener el máximo de throughput en saturación, $\overline{X}_C \approx 1$, a costa de eliminar, en la práctica, los periodos de *vacation*. El modelo M1 empieza a saturarse a partir del valor $\overline{\lambda} \approx 0.9$. Si se compara con los datos de laboratorio, sólo se ajusta para valores por debajo de $\overline{\lambda} \approx 0.8$. Por su parte, el modelo M2, a diferencia de M1, sí se ve afectado por los periodos de inactividad y el throughput de captura disminuye. El caso M2_V1 se ajusta a los valores de laboratorio en su totalidad. Alcanza el máximo para $\overline{X}_C \approx 0.8$. El decrecimiento posterior se debe al aumento de los tiempos de *vacation* y a la finalización de la *softirq* por *budget*. Para las tasas más altas, $\overline{\lambda} > 1$, el throughput se vuelve aproximadamente constante debido a que los tiempos medios de *vacation* se estabilizan y la duración del proceso de captura viene fijada por el límite B. Los casos M2_V2 y M2_V3 sirven para predecir el comportamiento del sistema en otros escenarios. Se ve que la forma es similar a la de M2_V1, pero se alcanzan valores de throughput menores.

A continuación, la Fig. 5b muestra la variación del throughput con respecto al *budget*, B, sobre el escenario V1. Se mantiene $N = 200$. El modelo M1 no tiene definido propiamente el parámetro *budget*, pero puede considerarse equivalente al caso extremo $B \to \infty$. El caso M2_B300 se ajusta a los valores de laboratorio. Con valores de *budget* menores, casos M2_B200 y M2_B100, se observa que el throughput en la zona de saturación ($\overline{\lambda} > 0.8$) disminuye, tanto más cuanto menor sea el valor de B. Esto se debe a que, cuando se agota el *budget*, el tiempo dedicado a la captura es menor cuanto menor es B. Si se evalúa el modelo M2 para valores de $B > 300$, se obtienen throughputs mayores al caso M2_B300, pero siempre por debajo del caso extremo M1.

Por su parte, la Fig. 5c muestra la variación de la frecuencia de *softirq*. Este valor pueda dar idea del número de cambios de contexto que se dan entre los periodos de captura y *vacation*. En los tres escenarios (V1, V2 y V3) y para ambos modelos (M1 y M2), se pueden distinguir tres zonas: una para tasas bajas, en la que a medida que aumenta la tasa de entrada, y con ella la

actividad de captura, el número de *softirq* por segundo crece hasta que llega a un punto máximo. A partir de ahí, con tasas de entrada intermedias, debido al aumento de los tiempos de *vacation*, la frecuencia de *softirq* disminuye. Finalmente, hay una tercera zona para tasas de entrada más altas donde, en el caso del modelo M1, la frecuencia de *softirq* disminuye porque la duración de la *softirq* se alarga indefinidamente, no existen prácticamente *vacations* y $f_{sirq} \to 0$; por contra, en la tercera zona del modelo M2 la frecuencia de *softirq* se mantiene constante, con un valor bajo, ya que la duración media de la *softirq* toma el valor B/μ_S. En la Fig. 5c también se da que el caso M2_V1 se ajusta a la medida de laboratorio.

También se ha analizado los resultados de frecuencia de *hardirq*, f_{hirq} en el modelo M2. Se ha comprobado que, para tasas de entrada en las que no se alcanza el *budget*, $f_{hirq} \approx f_{sirq}$; y que, para tasas en la zona de saturación, se llega a un valor extremo tal que $f_{hirq} \to 0$.

Por último, la Fig. 5d presenta el número medio de paquetes por *softirq*, definido como $\overline{m} = \mu_S T_{sirq}$. Permite visualizar, a partir de qué tasa de entrada, la ejecución de la *softirq* alcanza el valor del *budget* ($B = 300$ en la gráfica). Obviamente, el escenario V3 es el que agota el *budget* con menor tasa de entrada, y el escenario V1 el que lo alcanza con mayor tasa. Una vez más, el caso M2_V1 es el que se ajusta a los valores medidos en la sonda de laboratorio.

Cabe mencionar que también se han evaluado los modelos con tamaños de búfer mayores que 200 (en concreto, $N = 512$), pero los resultados obtenidos han sido similares. También se ha probado qué ocurre si se desprecian las *hardirq* en el modelo M2 (suponiendo $1/\mu_H \to 0$) y la conclusión ha sido que es factible esa suposición, ya que los resultados obtenidos son prácticamente iguales.

V. CONCLUSIONES

Este trabajo plantea la evaluación del rendimiento de un sistema de captura de paquetes a partir de un modelado analítico consistente en un sistema de cola con *vacations*. Este concepto permite modelar el comportamiento del procesador responsable de capturar paquetes y, a su vez, de realizar tareas adicionales en los denominados tiempos de *vacation*.

M1, el primer modelo propuesto, puede considerarse un modelo simple que caracteriza a un sistema de captura genérico que, dada la disciplina de servicio exhaustiva, prioriza el proceso de captura. En condiciones en las que el sistema no está saturado, garantiza la ejecución de tareas adicionales en los denominados tiempos de *vacation*; por el contrario, en condiciones de saturación, el proceso de captura acapara el tiempo del procesador en detrimento de realizar otras tareas adicionales, pero se consiguen throughput de captura aceptables .

M2, el segundo modelo, es más complejo y su disciplina de servicio limitada recoge las particularidades de sistemas de captura como Linux que establecen un límite para la ejecución del proceso de captura. Esto garantiza la ejecución de otras tareas para todos los casos, pero conlleva una pérdida de throughput de captura. En estos casos será interesante valorar el beneficio que supone disponer de tiempos de *vacation* para dichas tareas adicionales, frente a la pérdida potencial de throughput de captura.

Las conclusiones de este trabajo son satisfactorias en lo que respecta al comportamiento de los modelos. No obstante, este trabajo, que combina estudio analítico con medidas experimentales, trae varios aspectos a considerar en un futuro próximo. En primer lugar, se considera interesante estudiar modelos con *vacations* con procesos que no sean de tipo Poisson. En estos casos, si la resolución analítica se vuelve intratable, no se descarta la opción de evaluar el rendimiento mediante técnicas de simulación. Por otro lado, dado que, en un sistema de monitorización de tráfico de red, la etapa de captura puede estar relacionada con tareas posteriores de análisis o de otra índole, puede ser susceptible de estudio el modelado y la evaluación del rendimiento del sistema completo de monitorización de red, teniendo presente la influencia del planificador de tareas.

REFERENCIAS

[1] N. Tian, Z.G. Zhang, "Vacation Queuing Models. Theory and Applications", Springer Science+Business Media, 2006.

[2] H. Takagi, "Queueing Analysis. A Foundation of Performance Evaluation Volume 1: Vacation and Priority Systems (Part 1)", North-Holland, 1991.

[3] S. Lee, K. Levanti, H.S. Kim, "Network monitoring: Present and future", Computer Networks, vol. 65, n. 2, pp. 84-98, 2014.

[4] B. Li, J. Springer, G. Bebis, M. H. Gunes, "A survey of network flow applications", Journal of Network and Computer Applications, vol. 36, n. 2, pp. 567-581, 2013.

[5] S. Gallenmuller, P. Emmerich, F. Wohlfart, D. Raumer, G. Carle, "Comparison of frameworks for high-performance packet IO", Proceedings 2015 ACM/IEEE Symposium on Architectures for Networking and Communications Systems, pp. 29-38, 2015.

[6] ntop project, http://www.ntop.org

[7] S. Han, K. Jang, K. Park, S. Moon "PacketShader: a GPU-accelerated software router", ACM SIGCOMM Computer Communication Review, vol. 41, n. 4, pp. 195-206, 2011.

[8] L. Rizzo, "netmap: A Novel Framework for Fast Packet I/O", USENIX Annual Technical Conference, pp. 101-112, 2012.

[9] N. Bonelli, A. Di Pietro, S. Giordano, G. Procissi, "On multi-gigabit packet capturing with multi-core commodity hardware", Proceedings Passive and Active Measurement, pp. 64-73, 2012.

[10] OpenOnload, http://www.openonload.org/

[11] V. Moreno, P.M. Santiago del Río, J. Ramos, J.L. García-Dorado, I. González, F.J. Gómez, J. Aracil, "Packet Storage at Multi-gigabit Rates Using Off-the-Shelf Systems", 16th IEEE International Conference on High Performance and Communications, 2014.

[12] W. Wu, M. Crawford, M. Bowde, "The performance analysis of Linux networking - packet receiving", Computer Communications, vol. 30, n. 5, pp. 1044-1057, 2007.

[13] K. Salah, K. El-Badawi, R. Boutaba, "Performance modeling and analysis of network firewalls", IEEE Transactions on Network and Service Management, vol. 9, n. 1, pp. 12-21, 2012.

[14] G. Bolch, S. Greiner, H. Meer, K.S. Trivedi, "Queuing Networks and Markov Chains", John Wiley&Sons, 1998.

[15] I. Adan, J. Resing, "Queueing Theory", Eindhoven University of Technology, 2002.

[16] A. Muñoz, A. Ferro, F. Liberal, J. Lopez, "A Kernel-Level Monitor over Multiprocessor Architectures for High-Performance Network Analysis with Commodity Hardware" Proceedings SensorComm 2007 Valencia, 2007.

[17] A. Pineda, L. Zabala, A. Ferro, "Network Architecture to Automatically Test Traffic Monitoring Systems", Mosharaka International Conference on Communications and Signal Processing, 2012.

[18] D.P. Bovet, M. Cesati, "Understanding the Linux Kernel, Third Edition". O'Reilly Media, 2005.