eman ta zabal zazu

Universidad del País Vasco
Euskal Herriko Unibertsitatea

Departamento de Ingeniería de Comunicaciones
Escuela de Ingeniería de Bilbao

Tesis Doctoral

Modelos matemáticos basados en consumos computacionales para el estudio de rendimiento de sondas de análisis de tráfico en redes de datos

Autor:
Luis Zabala Alberdi

Director:
Dr. Armando Ferro Vázquez

Bilbao, Mayo de 2023

Gure Amari,
beti hor egoteagatik.

Familiari eta, dagoen
lekuan dagoela, Aitari.

UNIVERSIDAD DEL PAÍS VASCO -UPV/EHU-

Resumen

Escuela de Ingeniería de Bilbao
Departamento de Ingeniería de Comunicaciones

Doctor Ingeniero de Telecomunicación

Modelos matemáticos basados en consumos computacionales para el estudio de rendimiento de sondas de análisis de tráfico en redes de datos

por Luis Zabala Alberdi

La evolución de las redes de datos hacia velocidades más altas exige disponer de sistemas de monitorización y análisis de tráfico con mejores prestaciones. Por otro lado, en las redes actuales 5G y las propuestas de futuro B5G (Beyond 5G), con el auge de la virtualización, es común disponer de entidades software distribuidas por la infraestructura de red que deben realizar las funciones básicas de una sonda de tráfico como es el capturar y dedicar recursos al análisis de lo capturado. Este trabajo pretende realizar aportaciones proponiendo modelos matemáticos que representan este tipo de sistemas y evalúan su rendimiento. Los modelos toman como base los consumos computacionales del software implicado.

Antes de presentar las propuestas de modelado, se expone un estado del arte que describe la evolución de los sistemas de captura y análisis de tráfico de red, desde las primeras redes Gigabit Ethernet hasta la actual red 5G. Destaca el uso extendido de hardware de propósito general (commodity hardware) y de sistemas basados en Linux. También es de señalar que las propuestas más recientes se adaptan a las redes 5G, tratando a la monitorización como un servicio virtualizado más.

El primer modelo trata de resolver una problemática detectada en una sonda de análisis real desarrollada por el grupo de investigación NQaS. A través de unas medidas experimentales, se detecta un rendimiento bajo de la sonda, debido a un uso ineficiente de recursos computacionales. El modelo propuesto consiste en una cola tándem, de dos etapas, con un único servidor activo. En este entorno, se formula un proceso de decisión de Markov (MDP) tridimensional, con objeto de determinar la posición del servidor activo en cada intervalo de tiempo que considera el modelo, para que se mejore el throughput del sistema de colas. Como resultado del proceso MDP, se obtiene una política óptima cuya estructura se analiza para un amplio rango de escenarios. Se simula el resultado de la política óptima y se considera la implementación de los resultados teóricos sobre una sonda real.

El segundo modelo analiza en profundidad el sistema de captura de paquetes de Linux. Para centrarse en el sistema de captura, se propone

un modelo de colas con vacaciones o vacations. Este considera, por un lado, el proceso de captura y, por otro. el denominado tiempo de vacation. Se entiende como vacation al tiempo en el que el sistema no está activo con la tarea principal (la captura) y puede ocuparse de otras. En este segundo modelo se plantean tres variantes: a) distribuciones de tiempo exponenciales y valor infinito para el parámetro budget de softirq; b) distribuciones de tiempo exponenciales y valor finito para el parámetro budget; c) distribuciones de tiempo de tipo general y valor finito para el parámetro budget. Para resolver estos modelos se plantean cadenas de Markov en tiempo continuo, cadenas de Markov embebidas y la técnica de la variable suplementaria. Se resuelven los modelos y se obtienen métricas como throughput de captura, disponibilidad de CPU, frecuencias de hardirq, de softirq... La validación se lleva a cabo comparando con medidas experimentales de laboratorio.

Después de desarrollar estos dos modelos, se pretende ver cómo esta investigación previa, basada en teoría de colas, puede ser útil e integrable en el estudio de tecnologías de redes B5G. Por ello, se hace un estudio en el que se identifican posibles servicios 5G susceptibles de ser modelados y se propone el tercer modelo. Este plantea un problema de despliegue de funciones VNF (Virtual Network Function) en un escenario de servicios de Misión Crítica (MC) sobre una red B5G. El criterio de optimización es la latencia de las diferentes clases de tráfico; en concreto, se identifican cinco clases: señalización, floor control, voz PTT (Push To Talk), datos y vídeo. Para estimar esa latencia se propone un modelo basado en una red de colas M/M/1, donde cada cola representa a una VNF. Se plantea un problema de optimización que busca minimizar los tiempos de servicio, sujeto a una serie de restricciones de carácter computacional y energético que deben cumplir los nodos donde se pueden desplegar las VNF. Para resolver el problema de optimización, primero se reformula el problema para que sea convexo; luego, se sigue un método iterativo que busca posibles soluciones y se aplica la heurística denominada MaxZ. Para acabar, se presentan resultados del modelo para varios casos de estudio.

Abstract

The evolution of the data networks towards higher speeds demands the use of traffic monitoring systems with better capabilities. On the other hand, in the current 5G networks and the future Beyond 5G (B5G) networks, with the growth of virtualization, it makes sense to have available distributed software entities by the infrastructure of the network. Those entities can do the basic functions of a network traffic probe which is to capture and to dedicate resources to the analysis of what is captured. This work aims to make contributions based on mathematical models that represent this type of systems and evaluate their performance. The models consider the computational consumptions of the software involved.

Before presenting the modelling proposals, it is exposed a state of the art that describes the evolution of the network traffic capture and analysis systems, from the first Gigabit Ethernet networks until the current 5G networks. It highlights the extended use of commodity hardware and Linux based systems. Also, it is to show what are the most recent proposals that adapt to the 5G networks, taking into account the monitoring such as another virtualized service.

The first model is to solve the problem detected in a real probe developed by the research group NQaS. Through some experimental measures, a low performance of the probe is detected, due to an inefficient use of computational resources. The proposed model consists of a single-server tandem queue. In this framework, it is formulated a three-dimensional Markov Decision Process (MDP), with the aim to determine the position of the active server in every time slot that the model considers, so that the system throughput improves. As a result of the MDP process, an optimal policy is obtained and its structure is analysed for a wide range of scenarios. The result of the optimal policy is simulated and an implementation based on the theoretical results for a real probe is considered.

The second model analyses in depth the Linux packet capturing system. To focus on the capture system, it is proposed a queueing model with vacations. This takes into consideration, on the one hand, the capture process and, on the other hand, the denominated time of vacation. What is meant by vacation is the time that the system is not active with the principal task (capture) and it can deal with others. In this second model, three different variants are proposed: a) exponential time distributions and infinite value for the parameter budget of the softirq process; b) exponential time distributions and a finite value for the parameter budget; c) General-type

time distribution and a finite value for the parameter budget. To solve these models, continuous time Markov chains, embedded Markov chains and the supplementary variable technique are used. The models are solved and some metrics like capture throughput, CPU availability, hardirq and softirq frequencies...are obtained. The validation is completed comparing certain laboratory experiments.

After developing those two models, it is intended to see how this previous research, based on queueing theory, can be useful and integrable in the study of B5G network technologies. For that purpose, a study is made to identify what possible 5G services can be modeled and therefore the third model is created. This poses a problem of deploying Virtual Network Functions (VNFs) in the case of Mission-Critical services on the B5G network. The criteria of optimization is the latency of the different classes of traffic; specifically, five classes are identified: signaling, floor control, Push To Talk (PTT) voice, data and video. To estimate the latency, it is proposed a model based on a network of M/M/1 queues, where each queue represents a VNF. It is posed an optimization problem to find a way to minimise the service delays, subject to a series of computational and energy constraints that nodes on which VNFs can be deployed must comply. To solve the optimization problem, first, the problem is reformulated to make it convex; later, an iterative method is followed to search for possible solutions and the heuristic denominated MaxZ is applied. To finish, the results for various cases are presented.

Laburpena

Datu-sareen bilakaera gero eta abiadura handiagoetarantz doanez, beharrezkoa da trafikoa monitorizatzeko eta aztertzeko sistemak izatea, prestazio hobeekin. Bestalde, gaur egungo 5G sareetan eta etorkizuneko B5G (Beyond 5G) proposamenetan, birtualizazioaren gorakadarekin, ohikoa da sareko azpiegituran banatutako software-entitateak edukitzea, trafiko-zunda baten oinarrizko funtzioak bete behar dituztenak, hala nola trafikoa harrapatzea eta harrapatutakoa aztertzeko baliabideak eskaintzea. Lan honen bidez, mota horretako sistemak irudikatzen dituzten eredu matematikoak proposatu eta horien errendimendua ebaluatu nahi da. Ereduek inplikatutako softwarearen kontsumo konputazionalak hartzen dituzte oinarritzat.

Eredu-proposamenak aurkeztu aurretik, sareko trafikoaren harrapaketa-eta analisi-sistemen bilakaera deskribatzen duen artearen egoera azaltzen da, lehen Gigabit Ethernet sareetatik gaur egungo 5G sareraino. Helburu orokorreko hardwarearen (commodity hardware) eta Linuxen oinarritutako sistemen erabilera hedatua nabarmentzen da. Aipatzekoa da, halaber, proposamen berrienak 5G sareetara egokitzen direla, eta monitorizazioa zerbitzu birtualizatu gisa tratatzen dela.

Lehenengo eredua NQaS ikerketa-taldeak garatutako benetako analisi-zunda batean hautemandako problematika bat konpontzen saiatzen da. Neurri esperimental batzuen bidez, zundaren errendimendu baxua detektatzen da, baliabide konputazionalen erabilera ez-eraginkorraren ondorioz. Proposatutako eredua bi etapako tándem ilara bat da, zerbitzari aktibo bakarra duena. Ingurune horretan, hiru dimentsioko Markoven erabakitze-prozesu (Markov Decision Process-MDP) bat formulatzen da, ereduak aintzat hartzen duen denbora-tarte bakoitzean zerbitzari aktiboaren posizioa zehazteko, ilara-sistemaren throughput -a hobetu dadin. MDP prozesuaren ondorioz, politika ezin hobea lortzen da, eta horren egitura eszenatoki-maila zabal baterako aztertzen da. Politika optimoaren emaitza simulatzen da, eta emaitza teorikoak benetako zunda baten gainean inplementatzea hartzen da kontuan.

Bigarren ereduak sakon aztertzen du Linuxek paketeak harrapatzeko duen sistema. Harrapaketa-sisteman zentratzeko, oporrak edo vacations-ak dituen ilara-eredu bat proposatzen da. Alde batetik, harrapaketa-prozesua hartzen du kontuan, eta, bestetik, vacation-denbora deritzona. Vacation-tzat hartzen da sistema zeregin nagusiarekin (harrapaketa) aktibo ez dagoen eta beste batzuez arduratu daitekeen denbora. Bigarren eredu honetan hiru aldaera planteatzen dira: a) denbora-banaketa esponentzialak eta

softirq-aren budget parametrorako balio infinitua; b) denbora-banaketa esponentzialak eta budget parametrorako balio finitua; c) denbora-banaketa orokorrak eta budget parametrorako balio finitua. Eredu horiek ebazteko, Markov-en kateak planteatzen dira denbora jarraituan, Markov-en kate enbebituak eta aldagai osagarriaren teknika. Modeloak ebazten dira eta metrikak lortzen dira, hala nola harrapaketa throughput-a, CPU-aren eskuragarritasuna, hardirq eta softirq frekuentziak.... Balidazioa laborategiko neurri esperimentalekin alderatuz egiten da.

Bi eredu horiek garatu ondoren, aurretiko ikerketa hori, ilara-teorian oinarritua, baliagarria eta B5G sareen teknologien azterketan integragarria izan daitekeela ikusi nahi da. Horregatik, azterlan bat egiten da modelatu daitezkeen 5G zerbitzu posibleak identifikatzeko, eta hirugarren eredua proposatzen da. Horrek B5G sare baten gainean Misio Kritikoko zerbitzuen agertoki batean VNF (Virtual Network Functions) funtzioak hedatzeko kasua planteatzen du. Optimizazio-irizpidea trafiko mota desberdinen latentzia da; zehazki, bost mota identifikatzen dira: seinaleztapena, floor control, PTT (Push To Talk) ahotsa, datuak eta bideoa. Latentzia hori kalkulatzeko, M/M/1 ilara-sare batean oinarritutako eredu bat proposatzen da, non ilara bakoitzak VNF bat ordezkatzen duen. Optimizazio-problema bat planteatzen da, zerbitzu-denborak minimizatzea bilatzen duena, VNFak hedatzeko nodoek bete behar dituzten konputazio- eta energia-arloko murrizketa batzuen mende. Optimizazio-problema ebazteko, lehenik eta behin problema berriro formulatzen da ganbila izan dadin; gero, balizko soluzioak bilatzen dituen iterazio-metodo bati jarraitzen zaio, eta MaxZ izeneko heuristika aplikatzen da. Amaitzeko, ereduaren emaitzak aurkezten dira zenbait adibide-kasutarako.

Agradecimientos

En primer lugar, quiero agradecer infinitamente a mi Director de Tesis, Armando, por apoyarme siempre, tanto en este largo camino que he tomado para terminar los estudios de doctorado, como en los demás momentos que compartimos en la Escuela y fuera de ella. Siento que mis palabras se queden cortas para expresar todo lo que debo agradecerte, Armando.

También quiero acordarme de todos aquellos que han colaborado conmigo o han tenido simples muestras de apoyo. Empezando por mis compañeros del grupo "Networking, Quality and Security (NQaS)", gracias a mi querida Eva, a Alex (me he acordado mucho de él estos días), a mi (no menos querida) compañera de despacho Cristina (gracias por compartir mis silencios), a Ianire, a Bego, a Nekane, a Leire, a Aitor y, cómo no, a nuestro líder Fidel. También al grupo, como entidad que ha financiado mi asistencia a congresos internacionales, PhD Schools y otros gastos relacionados con esta Tesis.

Siguiendo con mis compañeros de departamento, quiero acordarme de Juanjo U. (gracias por no dejar de insistir con la pregunta), Juanjo I., Eduardo, Koldo y Mariví.

Gracias también a los exalumnos que, en su día, estuvieron trabajando dentro del grupo de investigación y tuvieron más relación directa conmigo como Alberto P., Ander N., Rubén S., Dani F., Lander A. y Javi D.

También quiero nombrar expresamente a Josu Doncel, por haberme impulsado en este sprint final.

Azkenik, familiari eskerrak eman nahi dizkiot eta lan hau berari eskaini; guztien gainetik gure Amari, beti niregandik hurbil eta nigatik edozer egiteko prest egoteagatik. Ez ditut aipatu gabe utzi nahi ere nire anaiarrebak, Agustin eta Ana, eta Aita.

Bilbao, Mayo de 2023

Índice de Contenidos

Índice de Figuras

Acrónimos y Abreviaturas

3GPP	**T**hird **P**artnership **P**roject
4G	**C**uarta **G**eneración
5G	**Q**uinta **G**eneración
6G	**S**exta **G**eneración
AI	**A**rtificial **I**ntelligence
AS	**A**pplication **S**ervices
AMF	**A**ccess and **M**obility Management **F**unction
API	**A**pplication **P**rogramming **I**nterface
AUSF	**Au**thentification **S**erver **F**unction
B5G	**B**eyond **5**G
C-RAN	**C**loud-**RAN**
CE-RAN	**C**loud **E**nabled **RAN**
CF	**C**ontrolling **F**unction
CMS	**C**onfiguration **M**anagement **S**erver
COTS	**C**ommodity-**O**ff-**T**he-**Shelf**
CPU	**C**entral **P**rocessing **U**nit
CUPS	**C**ontrol and **U**ser **P**lane **S**eparation
D2D	**D**evice-**T**o-Device
DevOps	Software **Dev**elopment and IT **Op**erations
DMA	**D**irect **M**emory **A**ccess
DMR	**D**igital **M**obile **R**adio
DPDK	**D**ata **P**lane **D**evelopment **K**it
DPMI	**D**istributed **P**assive **M**onitoring **I**nfrastructure
DR-GPS	**D**ominant **R**esource **G**eneralized **P**rocessing **S**haring
E2E	**E**nd-**T**o-**E**nd
eMBB	**e**nhanced **M**obile **B**road**b**and
EPC	**E**volved **P**acket **C**ore
ETSI	**E**uropean **T**elecommunications **S**tandards **I**nstitute
FPGA	**F**ield-**P**rogrammable **G**ate **A**rray

Gbps	**G**iga**b**it **p**or **s**egundo
GMM	**G**aussian **M**ixture **M**odel
GMS	**G**roup **M**anagement **S**erver
GPS	**G**lobal **P**ositioning **S**ystem
GPU	**G**raphical **P**rocessing **U**nit
GSPN	**G**eneralized **S**tochastic **P**etri **N**ets
HD	**H**igh **D**efinition
HSS	**H**ome **S**ubscriber **S**erver
IA	**I**nteligencia **A**rtificial
IDMS	**Id**entity **M**anagement **S**erver
IEEE	**I**nstitute of **E**lectrical **a**nd **E**lectronics **E**ngineers
IF	**I**mpact **F**actor
IMS	**I**P **M**ultimedia **S**ubsytem
IoT	**I**nternet **o**f **T**hings
IP	**I**nternet **P**rotocol
IRQ	**I**nterruption **Req**uest
ISR	**I**nterruption **S**ervice **R**outine
ITU	**I**nternational **T**elecommunications **U**nion
IWF	**I**nter**w**orking **F**unction
JCR	**J**ournal **C**itation **R**eports
KMS	**K**ey **M**anagement **S**ervice
KPI	**K**ey **P**erformance **I**ndicator
KVM	**K**ernel-based **V**irtual **M**achine
LTE	**L**ong **T**erm **E**volution
MaaS	**M**onitoring **a**s **a** **S**ervice
MAC	**M**edium **A**ccess **C**ontrol Layer
MC	**M**isión **C**rítica
MCPTT	**M**ission **C**ritical **P**ush **T**o **T**alk
MCPTT AS	**MCPTT A**pplication **S**erver
MCX	**M**ission-**C**ritical Services
MDP	**M**arkov **D**ecision **P**rocess
MEC	**M**ulti-Access **E**dge **C**omputing
mIoT	**m**assive **I**nternet **o**f **T**hings
ML	**M**achine **L**earning
MME	**M**obility **M**anagement **E**ntity
mMTC	**m**assive **M**achine-**T**ype **C**ommunications
MTC	**M**achine **T**ype **C**ommunications
MVA	**M**ean **V**alue **A**nalysis
NAPI	**N**ew **API**
NRF	**N**etwork **R**epository **F**unction
NFV	**N**etwork **F**unctions **V**irtualization

NIC	**N**etwork **I**nterface **C**ard
NTP	**N**etwork **T**ime **P**rotocol
NUMA	**N**on-**U**niform **M**emory **A**ccess
NQaS	**N**etworking, **Q**uality and **S**ecurity research group
O-RAN	**O**pen **RAN**
PDN-GW	**P**acket **D**ata **N**etwork **G**ateway
PF	**P**articipating **F**unction
PNF	**P**hysical **N**etwork **F**unction
PHY	**PHY**sical Layer
pps	**p**aquetes **p**or **s**egundo
QNA	**Q**ueueing **N**etwork **A**nalyzer
QoE	**Q**uality **o**f **E**xperience
QoS	**Q**uality **o**f **S**ervice
RAM	**R**andom **A**ccess **M**emory
RAN	**R**adio **A**ccess **N**etwork
RL	**R**einforcement **L**earning
RLC	**R**adio **L**ink **C**ontrol
RTCP	**R**eal **T**ime Transport **C**ontrol **P**rotocol
RTP	**R**eal **T**ime Transport **P**rotocol
SDN	**S**oftware **D**efined **N**etworking
S-GW	**S**erving **G**ate**w**ay
SINR	**S**ignal to **I**nterference-plus-**N**oise **R**ate
SIP	**S**ession **I**nitiation **P**rotocol
SJIF	**S**cientific **J**ournal **I**mpact **F**actor
SMF	**S**ession **M**anagement **F**unction
SNS	**S**oftwarized **N**etwork **S**ervices
SPN	**S**tochastic **P**etri **N**ets
TAS	**T**raffic **A**nalysis **S**ystem
Tbps	**T**era**b**it **p**or **s**egundo
TCP	**T**ransport **C**ontrol **P**rotocol
TIC	**T**ecnologías de la **I**nformación y la **C**omunicación
UDM	**U**nified **D**ata **M**anagement
UE	**U**ser **E**quipment
UPF	**U**ser **P**lane **F**unction
UPV/EHU	**U**niversidad del **P**aís **V**asco / **E**uskal **H**erriko **U**nibertsitatea
uRLLC	**u**ltra-High **R**eliability & **L**ow **L**atency **C**ommunications
VANET	**V**ehicular **A**d Hoc **Net**work
vMME	**v**irtual **M**obility **M**anagement **E**ntity
VNF	**V**irtual **N**etwork **F**unction
VNP	**V**irtual **N**etwork **P**robe

Capítulo 1

Introducción

En el desarrollo de las tecnologías para redes celulares se están produciendo muchos cambios desde las redes 4G hasta las actuales 5G y las propuestas de futuro más allá de 5G o *Beyond 5G (B5G)*. Estos últimos años el grupo de investigación *Networking, Quality and Security (NQaS)* ha desarrollado conocimiento en cómo afecta el incremento de volumen de tráfico en estas redes sobre las necesidades de consumo computacional para cursar servicios sobre ellas, y también en cómo pueden verse afectadas las comunicaciones, en concreto su calidad de servicio, en función de la disponibilidad de recursos computacionales para atenderlas. Estos dos aspectos son fundamentales en la evolución de las redes de nueva generación. A partir de 5G y en la provisión de servicios de las redes de futuro, aparece otro factor relevante; la propia red tiene inteligencia para proporcionar servicios de procesamiento de forma distribuida y surge la necesidad de definir estrategias para provisionar los servicios de forma adecuada, teniendo en cuenta la ubicación de los mismos en la infraestructura de la red. El conocimiento de las necesidades computacionales para cada servicio, el impacto que puede tener, en la calidad de servicio, la falta de recursos y la capacidad de ubicar esos recursos de forma flexible en la infraestructura de la red, son cuestiones muy relevantes para ayudar a diseñar y planificar mejor la provisión de servicios en las redes del futuro. La introducción en esta línea de investigación de la inteligencia de red en los algoritmos de análisis de tráfico y la asignación de recursos computacionales para la mejora de la provisión de los servicios es fundamental. Las técnicas de virtualización y despliegue de servicios sobre la infraestructura de las redes de futuro son también muy relevantes.

Inicialmente una sonda de análisis de tráfico consistía en un equipo que recogía tráfico de la red y realizaba una función determinada sobre ese tráfico, de interés en la gestión de la propia red. Se trata de sistemas de monitorización de tráfico para medir la calidad de servicio, de sistemas de detección de ataques basados en el tráfico de la red, de equipos de rutado de tráfico o pasarelas, etc. Sin embargo, con la evolución de las redes y la aparición de las tecnologías de virtualización, este tipo de sondas, hoy en

día, se pueden asociar a infinidad de sistemas que, de forma independiente, realizan funciones diversas en un entorno colaborativo. Los organismos de estandarización han ido realizando propuestas de mejora de las redes que introducen el uso de nuevas tecnologías como la virtualización de los elementos de red, la separación en funciones independientes del procesamiento de los flujos de tráfico y el despliegue distribuido de estas funciones en la infraestructura de la red. Cada uno de estos elementos abstractos que se han definido en los estándares pueden ser virtualizados y se pueden considerar como sondas de análisis de tráfico. En todos ellos es necesario capturar tráfico y realizar alguna función de análisis asociada a la petición que se curse. Originalmente las sondas de tráfico se centraban en gestionar el tráfico de datos de los usuarios, pero, en las nuevas redes, aparecen necesidades asociadas a la gestión del tráfico de señalización. Surge así la necesidad de caracterizar este tipo de sistemas.

El trabajo que se presenta en esta Tesis Doctoral estudia, en primer lugar, cómo caracterizar matemáticamente una sonda de tráfico, considerando los problemas que surgen en muchas de ellas referentes al potencial bloqueo del sistema de captura que se utilice, y en otros casos, buscando cómo optimizar su rendimiento considerando el coste computacional asociado al análisis de tráfico que puede comprometer la estrategia de captura. En segundo lugar, se plantea un modelo matemático basado en redes de colas, que permite estudiar cómo se puede desplegar de forma óptima una arquitectura distribuida para el análisis de tráfico basada en la virtualización de funciones de sondas de tráfico, para un servicio de referencia para misión crítica en redes 5G, como es el servicio *MCX (Mission-Critical Services)*. Bajo el término *MCX* se agrupan una serie de servicios de misión crítica que proporcionan comunicaciones de voz, datos y vídeo.

1.1. Contexto de la tesis

La trayectoria de investigación de esta tesis ha sido muy dilatada en el tiempo. Nace como continuación del trabajo de Tesis Doctoral "Propuestas de diseño de un sistema de detección de intrusión y definición de un modelo analítico para arquitecturas multiprocesador" del director de esta tesis, que se presentó en el año 2002. En las líneas futuras de esa tesis, el autor proponía la adaptación de su diseño a un sistema de análisis pasivo de tráfico para medir la calidad de servicio o *Quality of Service (QoS)* en la red. Se pueden distinguir, en esa dilatada trayectoria, tres fases interesantes:

- **Desarrollo tecnológico:** los primeros años se dedicaron a desarrollar prototipos, basados en arquitecturas multiprocesador, para poder disponer de una sonda de análisis de tráfico propia que sirviese de plataforma de pruebas para estudiar el rendimiento de la misma con diferentes soluciones de diseño. Se desarrolla una sonda basada en procesamiento en el kernel denominada Ksensor y otra denominada Adviser con procesamiento colaborativo en área de usuario. Se identificaron problemas en la captura de tráfico que llevaban a una

situación de bloqueo y se observó un impacto importante de la carga de análisis en el rendimiento del sistema.

- **Desarrollo científico:** Tomando como base la experiencia adquirida en el diseño de las sondas de tráfico, se desarrollan modelos matemáticos que sirven para estudiar el efecto de bloqueo en el subsistema de captura de paquetes del kernel de Linux. Se propone un modelo que permite analizar el impacto de la carga de análisis en el rendimiento de la sonda y ayuda en la planificación del sistema para decidir, en cada momento, si es mejor capturar o analizar. También se propone un modelo para estudiar las problemáticas de sincronización entre instancias de captura y análisis.

- **Integración con las tecnologías de las redes B5G:** La investigación previa estaba muy centrada en las problemáticas del grupo en el diseño de su sonda de análisis de tráfico. En los últimos años, se ha hecho un esfuerzo importante por aprovechar la investigación previa en su aplicación en las redes celulares con tecnologías emergentes 4G, 5G y B5G. En esta infraestructura aparecen muchos elementos susceptibles de ser modelados como sondas de tráfico. Se ataca entonces el problema de cómo coordinar las diferentes funciones que hay que realizar en la provisión de un servicio en estas redes y ayudar a resolver el problema de ubicación de recursos.

Tradicionalmente el grupo NQaS ha contemplado tres líneas de actuación en su investigación: una ligada a las sondas de tráfico en segmentos de alta capacidad, otra al estudio de la calidad de servicio de las redes con una visión integral (QoX) y una tercera sobre la gestión de recursos en la distribución de servicios en redes de nueva generación.

Estas líneas han estado interconectadas y han ido evolucionando a lo largo del tiempo. Sobre todo la tercera, que ha crecido enormemente y que ha dado significancia muy relevante al grupo a nivel internacional. Actualmente esta línea se conoce como "Despliegue y orquestación inteligente de servicios en entornos 5G+/6G". Dado el éxito de esta línea de investigación, se ha hecho un esfuerzo muy importante en las demás líneas para apoyar en lo posible los avances en el marco de las redes 5G y B5G. En concreto, la tesis que se presenta ha orientado los resultados de su investigación hacia su aplicación en el contexto de las redes B5G.

1.1.1. Línea de investigación dentro del grupo NQaS

El trabajo de la tesis se enmarca dentro de otra de las líneas de investigación del grupo NQaS de la Universidad del País Vasco (UPV/EHU); concretamente, en la línea que actualmente se denomina "Captura y procesado de datos de gran volumen para servicios virtualizados en redes inteligentes 5G+/6G".

El objetivo principal de esta línea de investigación es analizar diversos problemas relacionados con el gran volumen de datos que se manejan en los servicios en red, debido al compromiso que debe haber entre la capacidad computacional del sistema de procesamiento de datos y la velocidad

de los datos recibidos en las redes modernas, que es cada vez más elevada. En las redes B5G, será la propia infraestructura de red la encargada de proveer mecanismos de procesamiento distribuidos para determinadas funciones de red virtualizadas, *Virtual Network Function (VNF)*. Esto permitirá mejorar la respuesta de determinados servicios al poder aproximarse la provisión del mismo a las zonas capilares de la infraestructura donde generalmente están localizados los usuarios finales. Por lo tanto, hay que considerar en estas redes la evolución hacia la virtualización de los servicios, la integración de modelos matemáticos que ayuden en la toma de decisiones y las capacidades actuales de orquestación de servicios en estas infraestructuras basadas en metodologías modernas tipo DevOps.

En las nuevas redes B5G se consigue aumentar la capacidad computacional dedicada a los servicios al ser la propia red la que es capaz de proveerlos. Además, para ello se pueden utilizar arquitecturas de computación muy flexibles y con grandes recursos donde se pueden virtualizar muchos servicios (incluso procesamientos hardware) en algoritmos software basados en contenedores de servicio ubicuos en la infraestructura. Por debajo se pueden aprovechar arquitecturas multiprocesador y/o distribuidas que permiten disponer de una elevada capacidad computacional para resolver diferentes tipos de servicios. Muchos de ellos tienen que ver con la virtualización de funciones de red, *Network Functions Virtualization (NFV)* y otros con la forma de ofrecer provisión de servicios de red de forma orquestada (MANO- Management and Orchestation). Es de entender que para realizar esto es necesario dotar al sistema de una inteligencia que sea capaz de identificar soluciones de optimización multivariable a la hora de decidir dónde ubicar las diferentes entidades de procesamiento (VNF) y de qué forma coordinar su ejecución basándose en contenedores virtuales que permita proveer los servicios de forma efectiva. En ese sentido, el uso de técnicas de Inteligencia Artificial (IA) o modelos basados en teoría de colas pueden resultar ventajosos toda vez que, previamente, siga siendo para ello necesario un trabajo de modelado de muchos de los elementos de la arquitectura. Por ejemplo, es de interés ser capaz de proporcionar modelos matemáticos que permitan estimar el consumo de carga computacional en determinadas funciones de red (VNF), como es la captura de datos o el análisis de los mismos considerando que muchas de estas funciones estarán virtualizadas en contenedores de servicio. Ello exigirá el desarrollo de prototipos funcionales que se puedan someter a estudio en laboratorio y que permitirá parametrizar modelos teóricos (basados en teoría de colas) que sirvan para ayudar en la planificación de los servicios de red.

1.1.2. Sondas de tráfico en redes de alta capacidad

En esta tesis se desarrolla investigación sobre monitorización y análisis de tráfico de cara a conseguir avanzar en el diseño de las sondas y obtener modelos matemáticos que permiten estudiar de forma teórica los rendimientos de estos dispositivos en función de sus características físicas. El método científico que se ha utilizado en el trabajo es el de la experimentación empírica sobre la base de prototipos desarrollados en laboratorio y

la modelización matemática de los fenómenos físicos más relevantes que permiten estudiar el rendimiento de estos sistemas. Se contrastan los resultados experimentales de laboratorio con los resultados de los modelos matemáticos teóricos.

No se pueden estudiar todos los problemas, pues ello exige el desarrollo de muchos prototipos específicos para cada cuestión. Por ello, la acción se ha centrado fundamentalmente en analizar los problemas relacionados con las restricciones del sistema en cuanto a capacidad computacional, impacto de la carga de análisis en la captura y la sincronización de los procesos a planificar. Se estudian también los problemas de sincronización entre instancias de análisis y los que pueda haber también con los procesos de captura. Para ello se propone la utilización de modelos basados en redes de Petri.

El objetivo principal de esta investigación es identificar la forma más eficiente para analizar el tráfico en una red de alta velocidad (10Gps o superior) desde un equipamiento informático básico y poder realizar un tratamiento en tiempo real de ese flujo de datos pensando en diferentes aplicaciones, como sistemas de detección de intrusión, antivirus en red y monitorización de tráfico. El problema principal es disponer de suficiente capacidad de procesamiento, y saber aprovecharla para poder atender las necesidades de carga computacional que el análisis de ese tráfico precisa. El grupo de investigación NQaS está trabajando en la identificación de métodos y algoritmos para el análisis de tráfico en redes de datos de alta capacidad, en el diseño de sistemas de captura propios y en la creación de modelos matemáticos que ayuden a estimar el compromiso computacional del sistema en función de la carga de análisis asociada al tráfico.

Para alcanzar ese objetivo de investigación, las acciones que se han llevado a cabo han ido dirigidas en dos sentidos:

1. Mejora de la arquitectura de la sonda. Para esto, se realizan modificaciones de diseño específicas para mejorar el rendimiento del software. También se aprovechan las ventajas tecnológicas que puede ofrecer la evolución del hardware de propósito general.

2. Utilización de modelado teórico para predecir el rendimiento de la sonda en diferentes condiciones de tráfico de red y de carga computacional requerida para realizar el análisis de datos. Es aquí donde está el origen de este trabajo de tesis.

El grupo ha contado con experiencia reconocida a nivel nacional e internacional. A nivel nacional, participó en los grupos técnicos GT3 de control de la calidad en Internet del Ministerio de Industria, Energía y Turismo. El grupo formó parte de las redes temáticas MPLS y FIERRO (*Future Internet: Efficiency in high-speed networks*). A nivel internacional, el grupo contribuyó en actividades europeas de impulso del I+D+i como son las plataformas NESSI y NEM, la Asamblea sobre el Futuro de Internet (FIA) y el COST europeo denominado TMA (*Traffic Monitoring and Analysis*). Además, desde mayo de 2011, el grupo NQaS representa a la UPV/EHU como miembro académico de los foros de estandarización de la ITU-T por

invitación recibida de ese organismo con motivo de la relevancia de los trabajos de investigación desarrollados.

Entornos de aplicación de las sondas de análisis de tráfico

La investigación en el campo de la monitorización de tráfico en redes de telecomunicación ha adquirido gran relevancia en los últimos años. Para asegurar la disponibilidad de los servicios de red con el nivel de calidad requerido, es importante conocer continuamente el estado de los elementos de red (routers, switches, firewalls, servidores, máquinas virtuales...), medir su rendimiento y analizar el tráfico con objeto de detectar posibles anomalías, fallos de configuración, ataques o intrusiones. Hoy en día, la monitorización o medida de los parámetros de funcionamiento de una red es un proceso crítico que es necesario realizar en muchas aplicaciones del ámbito de las Tecnologías de la Información y la Comunicación (TIC). Por lo tanto, para ello es necesario poder desplegar herramientas y equipos de monitorización que obtengan métricas de rendimiento de red como latencia, jitter, pérdida de paquetes, throughput[1], número de paquetes duplicados, etc. Para realizar esta función se utilizan sondas para el análisis de tráfico que son un elemento importante en la caracterización de las redes.

Algunos de los entornos de aplicación más comunes de las sondas de análisis de tráfico son los siguientes:

- La ingeniería de tráfico, que implica monitorizar la red para obtener parámetros de rendimiento y calidad de servicio. Esto permite analizar la evolución del tráfico y detectar posibles sobrecargas en la configuración de la red. De esta manera, se pueden prever situaciones de congestión y disponer los recursos de una manera adecuada. Esto es importante para la propia operación de la red, pero también de cara a los usuarios, proporcionando indicios acerca de la calidad del servicio ofrecida con respecto a lo acordado o contratado con el proveedor.

- Los sistemas de seguridad. La detección de ataques o de intrusión, la previsión de los mismos y los sistemas antivirus precisan de elementos que procesen el tráfico en busca de patrones de ataque conocidos u operaciones sospechosas. La rápida detección de los ataques puede permitir poner en marcha respuestas adaptativas en los elementos de red o almacenar trazas de los ataques sufridos a modo de prueba para hipotéticas acciones legales.

- La gestión de la red. Esta no solo incluye aspectos de monitorización del funcionamiento de los segmentos de red y de los equipos, sino también recursos para facilitar la facturación de los servicios proporcionados.

[1]Se define el throughput como el número medio de paquetes por unidad de tiempo que el sistema puede tratar.

- La investigación. Un conocimiento adecuado del funcionamiento de las redes de datos es básico para que los investigadores pongan en marcha nuevas y mejores soluciones tecnológicas. Los resultados del análisis del tráfico son el parámetro de entrada para diseñar simulaciones, predicciones y modelos teóricos de comportamiento lo más precisos posible.

1.1.3. Despliegue y orquestación inteligente de servicios en entornos 5G y B5G

En esta área de trabajo de enfoque tecnológico, la investigación se centra en el desarrollo de nuevas herramientas inteligentes de orquestación de servicios en entornos virtualizados sobre arquitecturas de red 5G, B5G y 6G, para el despliegue de verticales industriales estratégicos. Sus entornos de aplicación prioritarios son la Industria 4.0 y el despliegue de aplicaciones de misión crítica sobre redes 5G, con el propósito de reforzar las actividades de transferencia del grupo de investigación.

La virtualización de muchos de los aspectos de las redes 5G representa en sí una oportunidad sobre la cual implementar soluciones. El aprovechamiento efectivo de estas soluciones requiere de su integración en un ecosistema de virtualización de funciones de red o *Network Functions Virtualization (NFV)* y de una operación y un control bajo redes definidas por software o *Software Defined Networking (SDN)*. Adicionalmente, un aspecto clave será el despliegue de estos servicios sobre unos recursos de computación en el extremo (*edge computing*).

En los últimos años, el grupo NQaS se ha especializado en servicios de Misión Crítica (MC). En situaciones de emergencia (naturales o provocadas por personas) y de misión crítica, los efectivos que acuden en primer lugar (militares, bomberos, policía, sanitarios, ...) normalmente deben actuar en un entorno difícil y no predecible. Sin lugar a dudas, en estas situaciones las comunicaciones y los sistemas de información resultan claves en la gestión de la crisis. En este contexto, el impulso de las redes de banda ancha 5G está generando una gran transformación en las comunicaciones de MC. El reto hoy consiste en integrar servicios de MC en el ecosistema 5G, permitiendo despliegues de servicios que comparten infraestructura. Existen distintos modelos de negocio y uno de ellos consiste en el despliegue de servicios MC sobre redes públicas coexistiendo con otros servicios comerciales. En concreto, el grupo de investigación ha adquirido un protagonismo especial en el área de servicios de misión crítica para comunicaciones Push to Talk, *Mission Critical Push To Talk (MCPTT)*.

Es en el campo de esos servicios de Misión Crítica (MC), donde se ha centrado en esta tesis el trabajo para aplicar el conocimiento adquirido en el modelado de sondas de análisis de tráfico aplicadas a entidades de gestión de tráfico multimedia de forma distribuida en la red B5G. Se considera viable la utilización de modelos matemáticos, basados en consumos computacionales para estimar el comportamiento de las entidades que se deben desplegar en este tipo de servicio para ofrecer determinadas funciones VNF que proveen el servicio MC. Se propone también en esta tesis un

modelo matemático para estudiar la ubicación óptima de esas funciones en la infraestructura disponible de la red B5G. Luego, la investigación en esta línea se va a centrar en los siguientes aspectos:

- Análisis del impacto en la virtualización de servicios en redes B5G para la captura y procesado de datos de gran volumen.
- Definición de estrategias inteligentes de ubicación de servicios en redes B5G en función de las necesidades computacionales en flujos de tráfico intenso

En el primero de los casos, se estudia el comportamiento de diferentes modelos de virtualización de servicios para procesar tráfico de alta velocidad utilizando sondas propias que pueden ubicar de formas diferentes las VNF. Para ello, se están desarrollando sondas propias que generan tráfico conforme susceptible de análisis. Estos experimentos ayudan a parametrizar entidades básicas de procesamiento que puedan ser utilizadas en los modelos matemáticos propuestos y que puedan ayudar en la planificación de estos servicios. Estos modelos ayudan a estimar el compromiso computacional del sistema en función de la carga de análisis asociada al tráfico para distribuciones de servicio concretas.

El segundo de los escenarios busca encontrar estrategias inteligentes de ubicación de los servicios para facilitar la provisión de servicios de datos en redes B5G. A partir de los resultados obtenidos en los modelos teóricos previos en escenarios concretos, se quiere investigar cómo los modelos de redes de colas propuestos pueden ayudar en la toma de decisiones para ubicar de la forma más eficaz posible las diferentes entidades de procesamiento virtualizadas (VNFs). Originalmente se planteaba el uso de técnicas de *Reinforcement Learning* en la estimación de recompensas, pero a partir de las investigaciones realizadas en esta tesis se ha visto más adecuado el uso de modelos basados en redes de colas que permiten modelar la cadena de provisión de las funciones virtualizadas de red.

1.2. Técnicas de evaluación de rendimiento

En los últimos años, la disponibilidad de hardware de captura de datos y de soluciones de almacenamiento masivo a un coste razonable ha hecho que surjan varias iniciativas de desarrollo de equipos, arquitecturas software, herramientas y métodos para la captura, análisis e interpretación del tráfico de las redes. Este trabajo de tesis pretende ayudar en el estudio de sondas de análisis de tráfico de red que puedan realizar labores de monitorización de una forma efectiva. Más concretamente, se espera que las aportaciones de esta tesis vayan orientadas al modelado teórico de este tipo de dispositivos desde el punto de vista del rendimiento computacional.

En general, el modelado de un sistema de telecomunicaciones consiste en obtener una representación simplificada del sistema bajo estudio (bien sea una red, un servicio, un protocolo o simplemente una parte que resulte interesante de analizar) que permita responder cuestiones sobre el mismo.

Si el modelo propuesto es de tipo matemático, un conjunto de expresiones matemáticas describen las relaciones existentes entre las magnitudes y atributos que caracterizan el sistema, así como el conjunto de reglas que determinan cómo dichos atributos interactúan [1]. Es importante no olvidar que las respuestas obtenidas a través del modelo siempre deben ir orientadas hacia los objetivos que se pretenden alcanzar. En el caso de este trabajo, se buscará que, a partir de los modelos matemáticos, se deriven métricas de rendimiento como:

- Throughput de la sonda de análisis de tráfico.
- Probabilidad de pérdida de paquetes.
- Tiempo de respuesta de la sonda.

Como se ha mencionado anteriormente, este trabajo de tesis se centra en la aplicación de modelos analíticos sobre equipos de monitorización de redes, con objeto de evaluar su rendimiento. Se propondrán diferentes modelos. En general, todos ellos pueden ayudar a resolver cuestiones de diseño, pero cada uno debe desarrollarse con sus características específicas: métricas de rendimiento, metodología a seguir y condiciones necesarias para realizar el análisis. Por ello, en cada modelo, siempre se establecerá una relación con las características concretas del sistema que se desea representar y evaluar. La literatura existente sobre técnicas de modelado que pueden aparecer en este trabajo es muy amplia y de calidad. No se pretende ahondar en ese terreno teórico, sino que se busca extraer conceptos ya definidos y aplicarlos en el contexto de interés del trabajo. Asimismo, cabe mencionar que los métodos que se presentarán aquí no son los únicos que se pueden utilizar para evaluar el rendimiento de sistemas de red. Hay varias técnicas y una amplia gama de herramientas asociadas. Cada una de ellas tiene sus pros y sus contras.

Las técnicas de evaluación de rendimiento se clasifican en tres categorías: modelado analítico, simulación y medición [2]. Empezando con las mediciones, estas solo son posibles si algo similar al sistema de estudio ya

Figura 1.1: Combinación de metodologías en el proceso de gestión del rendimiento.

existe, como por ejemplo cuando se está diseñando una versión mejorada de un producto. Sin embargo, si se tratara de un concepto nuevo, de las tres categorías mencionadas, solamente se podría elegir entre el modelado analítico y la simulación. Es decir, el modelado analítico y la simulación pueden ser usados en situaciones donde las mediciones no son posibles; sin embargo, en general, será un proceso más convincente aquel en el que el modelado analítico o la simulación utiliza datos de mediciones experimentales previas.

Por ejemplo, las medidas experimentales tomadas sobre una red real o el análisis de un sistema de logs puede proporcionar una información muy detallada del sistema real; sin embargo, esto requiere un gran esfuerzo en lo referente a consumo de recursos. Realizar un conjunto de pruebas sobre un sistema real puede necesitar una gran cantidad de configuraciones diferentes del equipamiento; esto supone un coste relativamente alto. Por el contrario, las fórmulas analíticas pueden proporcionar resultados para un conjunto amplio de escenarios diferentes, con un tiempo de configuración de parámetros muy pequeño y dando la posibilidad de analizar un espacio de soluciones mucho más rápidamente. No obstante, los métodos analíticos tienen el riesgo de hacer simplificaciones y suposiciones no muy realistas, produciendo resultados que podrían no ser como los del sistema real.

A pesar de existir diferencias, es posible que varias metodologías sean complementarias. Por ejemplo, se puede combinar aproximaciones analíticas con técnicas de simulación, o incluso con mediciones experimentales sobre redes reales. El objeto de ello será tener una representación del sistema real y una evaluación de rendimiento más afinadas. Aunque el trabajo de modelado ha sido el más relevante en esta tesis, también hay que destacar el uso de medidas experimentales y de resultados de simulación.

1.3. Objetivos

La evolución de las redes de datos, hacia velocidades cada vez más altas y con capacidad de ofrecer servicios virtualizados de computación, exige disponer de sistemas de monitorización y análisis de tráfico con mejores prestaciones. En las redes actuales, con el auge de la virtualización, es común disponer de entidades software dispersas por toda la infraestructura que actúan como pequeñas sondas de análisis de tráfico para poder realizar su función. No solo en lo que se refiere al flujo de datos de los usuarios, sino que también en la gestión de los flujos de datos de señalización. Basta observar los estándares desarrollados en las redes celulares para darse cuenta de las numerosas interfaces que se definen en estos protocolo y de la cantidad de funciones específicas definidas objeto de despliegue. Estas *minisondas* de tráfico de red realizan un procesado de los datos que puede ser enviado a una entidad de nivel superior o no; depende de la aplicación. En caso de que existan, las funciones de nivel superior se implementan en un conjunto reducido de equipos de gestión. Dichas funciones de nivel superior pueden ser, por ejemplo, el análisis de los datos agregados provenientes de las diferentes sondas o la presentación de alertas, avisos y otro tipo de información a las personas que operan sobre la red.

En este trabajo de tesis se estudia el modelado de sondas que capturan y analizan tráfico de red, con objeto de medir su rendimiento. Las tecnologías evolucionan y las características de los equipos cada vez son más completas, complejas y diversas. No es suficiente con utilizar sondas cada vez más potentes, ya que las necesidades computacionales de las nuevas redes de datos y los servicios que incorporan también son mayores. Además, en determinados casos, puede interesar adoptar soluciones más simples, donde los recursos disponibles (capacidad de procesamiento, memoria, etc.) pueden estar limitados. El uso de técnicas de modelado aplicadas a ese tipo de situación puede ser muy apropiado. Un modelo analítico puede predecir el rendimiento de una sonda con las condiciones de ese escenario de recursos limitados y determinar si puede responder de forma efectiva.

Con la aparición de las redes Gigabit y multi-Gigabit Ethernet, aparece el reto de obtener altos rendimientos en equipos de red que capturan y analizan paquetes. Surgen diversas soluciones para ello y su rendimiento se mide, en la mayoría de los casos, experimentalmente. En cambio, no son tantos los modelos matemáticos en este campo.

Por otro lado, la evolución hacia sistemas virtualizados hace que muchas de las funciones de red estén confinadas en pequeños contenedores software que se abstraen de las características físicas del sistema sobre el que funcionan, pero sí que deben realizar las funciones básicas de una sonda de tráfico como es el capturar y dedicar recursos al análisis de lo capturado.

Por ello, este trabajo de tesis pretende realizar aportaciones, proponiendo varios modelos que representan ese tipo de sistemas y permite evaluar su rendimiento bajo condiciones diversas (por ejemplo, diferentes tasas de tráfico de red o cargas de análisis variables, o diferentes mecanismos para la virtualización del servicio).

El **objetivo principal** de la tesis es avanzar en la investigación sobre modelos matemáticos aplicados sobre sistemas software de análisis de tráfico basados en arquitecturas computacionales diversas. Por un lado, se estudia el problema de implementación basado en arquitecturas hardware de propósito general, de cara a mejorar su rendimiento. Por otro, se adaptan los resultados previamente conseguidos a arquitecturas software virtualizadas para estudiar problemáticas de distribución del servicio. El proceso de modelado debe identificar elementos clave para optimizar el rendimiento del sistema. Sus resultados deben ayudar a encontrar opciones de diseño válidas que introduzcan mejoras o eliminen limitaciones en el despliegue de sondas que sean parte de un sistema distribuido para análisis de tráfico.

El enfoque investigador de este trabajo es más teórico que empírico. Se entiende que la parte empírica consiste en analizar en laboratorio el rendimiento de este tipo de sistemas, desarrollando para ello diferentes prototipos experimentales. No obstante, ambas vertientes, la teórica y la experimental, deben complementarse.

En la fase inicial de **Desarrollo tecnológico** el esfuerzo de investigación se centró en el desarrollo de prototipos funcionales de sondas de

análisis de tráfico con arquitecturas multiprocesador. Adviser es un prototipo funcional que permite capturar tráfico y analizarlo en espacio de usuario utilizando instancias de procesamiento colaborativas. Ksensor realiza esta función, pero en espacio del núcleo (kernel) del sistema operativo. Ambos prototipos han sido usados intensamente en esta fase de desarrollo tecnológico y han servido para identificar varios efectos de interés que han sido objeto de investigación en esta tesis.

- **Bloqueo en la captura de tráfico**. El rendimiento del sistema de captura se ve limitado por la usurpación de recursos que se produce en el subsistema de red en condiciones de tráfico elevado.
- **Impacto de la carga de análisis**. El rendimiento del sistema también se ve afectado por las necesidades computacionales que exige el proceso de análisis. Esto exige que en el diseño de una sonda sea importante establecer un balance entre el proceso de captura y las instancias de análisis.
- **Planificación de instancias de captura**. En arquitecturas multiprocesador es necesario decidir cómo organizar el proceso de captura en base a las instancias de ejecución disponibles y la forma de operar el subsistema de red utilizado.
- **Sincronización de instancias de análisis**. Al recibir los datos las instancias de análisis deben disponer de una información completa para poder hacer el seguimiento del tráfico. Hablamos de poder seguir conexiones o información de contexto que se debe compartir entre todas las instancias.

Posteriormente, en la fase de **Desarrollo científico** se centraron los trabajos en el desarrollo de modelos matemáticos que permitiesen modelar el comportamiento de estas sondas de cara a ayudar en la planificación de los sistemas de captura y análisis de tráfico. Se definieron algunos objetivos que han sido desarrollados en esta tesis y que a continuación se mencionan:

- Desarrollo de un modelo teórico basado en teoría de colas que permita demostrar el comportamiento de una sonda ante el problema de bloqueo del subsistema de red, en condiciones diferentes de intensidad de tráfico y carga de análisis.
- Desarrollo de un modelo teórico que ayude en estudiar el comportamiento de una sonda en el compromiso de dedicar su capacidad computacional a atender el proceso de análisis o a la captura de tráfico en una red con carga alta.
- Proponer algún modelo teórico que sea capaz de estudiar el conflicto de interacción entre instancias de captura considerando las secciones críticas en el proceso de captura.

En la fase final de **Integración con las tecnologías de las redes B5G** el trabajo consistía en ver cómo la investigación previa realizada podría ser útil en esas tecnologías y en identificar posibles escenarios de aplicación de los resultados. Esto exigía un estudio importante para analizar

el estado del arte del nuevo entorno y la asimilación de muchas tecnologías que se estaban desarrollando en el contexto de las redes 5G. Los objetivos de investigación en esta fase, desarrollados en esta tesis, han sido:

- Estudiar cómo afecta la virtualización de sistemas en la utilización de los modelos teóricos desarrollados en etapas anteriores.
- Proponer un modelo teórico basado en teoría de colas que permitiese evolucionar las propuestas previas del grupo NQaS. Estas propuestas se basaban en consideraciones de consumos de recursos muy deterministas y la teoría de colas puede proponer modelos más estocásticos que permitan considerar cierta aleatoriedad.
- Desarrollar un modelo teórico basado en redes de colas que pueda servir para estudiar los problemas de ubicación de recursos en la provisión de servicios en redes 5G.

El desarrollo de estos objetivos se condensa en la propuesta realizada en esta tesis sobre el modelado del servicio de Misión Crítica que pretende sustituir al servicio utilizado por bomberos, policía y otros servicios de emergencia en las redes 5G y que se está estandarizando en el 3GPP. Se trata del servicio Push to Talk que se conoce como MCPTT y su extensión MCX que contempla el envío, no solo de audio, sino también de vídeo y datos. En esta tesis se propone un modelo teórico basado en redes de colas que permite estudiar cómo puede desplegarse este tipo de servicios sobre una arquitectura 5G de forma distribuida para optimizar el uso de los recursos ofreciendo parámetros de calidad incluso mejores.

1.4. Estructura del documento

Este documento de tesis está organizado en seis capítulos. En este primer capítulo se ha pretendido contextualizar el trabajo realizado, presentando las diferentes fases en el desarrollo de la investigación. Se ha intentado explicar las problemáticas asociadas a las sondas de análisis de tráfico y sus posibles aplicaciones. Para ello, ha sido preciso describir brevemente las líneas de investigación donde se integra este trabajo de tesis. Se han identificado las técnicas más adecuadas para la evaluación del rendimiento presentado diferentes modelos de análisis que son objeto de estudio. Finalmente se han planteado los objetivos que se han definido para el trabajo de esta tesis, enmarcados dentro de los objetivos del grupo de investigación NQaS. En ellos se puede ver cómo, en una fase inicial, la investigación se ha centrado en el modelado de sistemas de análisis de tráfico y su rendimiento, para finalmente proponer un modelo que permite integrar esos trabajos dentro de las tecnologías de redes tipo 5G que están actualmente en auge.

A continuación, en el Capítulo 2 se presenta un estado del arte relacionado con los sistemas susceptibles de modelado en este trabajo. El capítulo se divide en dos bloques. En el primero, se presentan los sistemas de captura y análisis de tráfico de red, describiendo la evolución que han

tenido a lo largo del tiempo, desde las primeras redes Gigabit Ethernet hasta las actuales redes 5G y se hace especial hincapié en propuestas de arquitecturas software que utilizan hardware de propósito general. Este primer bloque también incluye una revisión de trabajos en los que se aplica modelado matemático sobre sistemas de análisis de tráfico en redes de datos.

El segundo bloque del Capítulo 2 aborda la identificación de servicios 5G susceptibles de ser modelados. Se plantean dos líneas de estudio como son la del modelado de una sonda B5G de captura y análisis de tráfico y la del modelado de la provisión de un servicio extremo a extremo en redes B5G. Dada la fuerte experiencia del grupo NQaS en la investigación sobre servicios de misión crítica, se opta por abordar la problemática de la distribución de este servicio virtualizado sobre una red B5G. Al igual que se hizo en el bloque anterior, este segundo bloque del Capítulo 2 se completa con una revisión de trabajos de modelado basados en teoría de colas aplicados sobre redes 5G.

Los Capítulos 3, 4 y 5 contienen la mayor contribución de este trabajo: los modelos matemáticos. En primer lugar, en el Capitulo 3, se describe una propuesta de modelado para optimizar el throughput de una sonda de análisis mediante la resolución de un proceso de decisión de Markov. Para ello, en primer lugar se describe la problemática observada en un prototipo de sonda de tráfico de red; posteriormente, se propone el modelo, que consiste en una cola tándem con un único servidor activo; a continuación, se plantea un proceso de decisión de Markov cuyo objetivo es determinar en cuál de las dos etapas de la cola tándem debe estar el servidor activo, de modo que se maximice el rendimiento del sistema; tras resolver numéricamente el modelo, se obtiene una política óptima que define la estrategia a seguir por el sistema; después, se evalúa el rendimiento que produce la política óptima con un conjunto de valores directamente relacionados con el entorno de investigación; para finalizar el proceso de modelado, se realiza la validación, primeramente, implementando sobre la sonda real una política no óptima pero sí próxima a la óptima y, en segundo lugar, mediante la simulación de una red de Petri estocástica generalizada.

El Capítulo 4 presenta la segunda propuesta de modelo donde se analiza el rendimiento de la etapa de captura de paquetes de una sonda. Para ello se propone un sistema de cola finita con *vacations*. El capítulo comienza con la descripción del sistema de captura que se va a modelar, el mecanismo del sistema operativo Linux denominado NAPI. Tras esta explicación, se propone el modelo de cola finita que incluye los procesamientos específicos de *hardirq* y *softirq* de Linux, así como el denominado tiempo de *vacation*, concepto que también es presentado en esta parte del capítulo. A continuación, se analizan tres casos (llamados M1, M2 y M3) del modelo genérico propuesto. El modelo M1, el más sencillo de los tres, supone tiempos exponenciales y parámetro *budget* (del proceso de recepción de paquetes de Linux) con valor infinito; el modelo M2 asume tiempos exponenciales y *budget* finito; por último, M3 es el modelo más complejo ya que contiene distribuciones de tiempo de tipo general y *budget* finito.

Tras resolver los modelos M1, M2 y M3, estos son evaluados y los resultados teóricos son contrastados con valores experimentales de una sonda real.

El Capítulo 5 contiene un modelo de red de colas para optimizar el despliegue de funciones VNF que proveen servicios de Misión Crítica sobre una red B5G. Para ello, se construye un modelo que contiene diferentes clases de tráfico (señalización, *floor control*, voz PTT, datos y vídeo) que son procesadas por diferentes funciones VNF. Para plantear el despliegue óptimo de las funciones VNF, el modelo de red de colas considera la latencia del servicio como parámetro clave. Por ello, se desarrolla la formulación de dicho retardo medio en una red de colas M/M/1 y se plantea un problema de optimización con una función objetivo que minimiza el retardo mencionado. Además deben cumplirse unas restricciones de capacidad computacional y consumo energético en los nodos en los que se despliegan las funciones VNF. Para finalizar el capítulo, se presentan varios casos de ejemplo donde se demuestra la utilidad del modelo propuesto.

Finalmente, el Capítulo 6 expone las conclusiones, incorporando a su vez un resumen de todo el trabajo realizado y planteando las cuestiones abiertas que constituyen las futuras líneas de investigación.

Capítulo 2

Estado del arte

Este capítulo contiene el estado del arte de los sistemas que son susceptibles de modelado en las propuestas que se expondrán en capítulos posteriores. Hay que tener en cuenta que el trabajo de esta tesis se centra inicialmente en cómo caracterizar matemáticamente una sonda de tráfico y, en segundo lugar, orientar el modelado utilizado en esa investigación, hacia entornos de redes 5G actuales y futuras B5G.

Se ha dividido el capítulo en dos bloques diferenciados. Los apartados 2.1 y 2.2 constituyen el primer bloque que está dedicado a los sistemas de captura y análisis de tráfico. En el apartado 2.1, primeramente, se explica por qué son necesarios este tipo de sistemas y cuáles son las etapas de un sistema de monitorización de red genérico. En segundo lugar, se hace hincapié en la arquitectura de referencia de una sonda de análisis de tráfico, que es un elemento dentro del sistema genérico anterior. Para terminar este primer apartado, se exponen diversas iniciativas (arquitecturas, herramientas y proyectos) de sistemas de analísis de tráfico que ha habido a lo largo del tiempo, desde los primeros años de las redes Gigabit Ethernet hasta las actuales redes de quinta generación (5G).

A continuación, en el apartado 2.2 se recoge la aplicación de modelos matemáticos basados en consumos computacionales sobre sistemas de captura y análisis de tráfico de red. Esto cierra el primer bloque.

El segundo bloque está formado por los apartados 2.3 y 2.4. Está relacionado con la otra línea de investigación de esta Tesis, la línea denominada "Despliegue y orquestación inteligente de servicios en entornos 5G, B5G y 6G".

En el apartado 2.3, con objeto de poder aplicar el conocimiento sobre modelado adquirido en la línea relacionada con las sondas de red, se analizan servicios de redes 5G que pueden ser objeto de modelado. Se identifican y se describen dos opciones: por un lado, una sonda de captura y análisis sobre red B5G y, por otro, la provisión de un servicio virtualizado extremo a extremo sobre red B5G.

Para cerrar el segundo bloque, el apartado 2.4 analiza trabajos previos relacionados con modelos de colas aplicados a entornos de redes 5G. El

objetivo es determinar si este tipo de modelado puede ayudar en ese nuevo escenario.

Finalmente, en el apartado 2.5, se recopilan brevemente las principales aportaciones estudiadas en los dos bloques anteriores y se extraen las conclusiones más relevantes de este estudio, a fin de orientar las posibles propuestas.

2.1. Sistemas de captura y análisis de tráfico de red

En los últimos años, son muy frecuentes las investigaciones que tienen como objetivo el desarrollo de nuevos sistemas de análisis de tráfico capaces de procesar toda información de interés que transportan las redes actuales. En este apartado, se hace una revisión de algunos de ellos, incidiendo especialmente en aquellos que están soportados sobre hardware de propósito general. Previamente, se explican algunos conceptos sobre sistemas de monitorización de tráfico y se especifica la arquitectura de referencia de una sonda de análisis de tráfico de red que va a ser objeto de estudio en este trabajo.

2.1.1. Sistemas de monitorización de red

La necesidad de monitorizar y analizar el tráfico de las redes de comunicaciones crece a medida que se incrementa el número de usuarios y los volúmenes de datos que se manejan. La llegada de la red 5G no ha hecho más que asentar la afirmación anterior. Con la ganancia de rendimiento que suponen las redes 5G, la cantidad de dispositivos conectados y conexiones establecidas se incrementa considerablemente. Según el informe "The Mobile Economy Report 2022" [3], las conexiones 5G superarán los 2000 millones en 2025. Uno de los motivos principales de este aumento es la heterogeneidad de los dispositivos conectados: desde ordenadores personales o servidores hasta dispositivos IoT (*Internet of Things*) utilizados en aplicaciones de telemedicina, vehículos autónomos o sensores de procesos industriales. También van por la misma línea los datos proporcionados por el informe "2023 Global Internet Phenomena Report" de Sandvine [4], que estima que hubo un incremento del 23 % en el volumen de tráfico de Internet en 2022.

Monitorizar una red es un proceso crítico del área de TIC. Disciplinas como la seguridad, el análisis de la calidad de servicio, la gestión de red, la facturación e incluso el encaminamiento requieren de sistemas de monitorización y análisis de tráfico de altas prestaciones. La monitorización de red guía a los operadores de red en el entendimiento del comportamiento de la red, para luego poder reconfigurarla y mejorar su estado. Una de las funciones más importantes de la monitorización de red es la identificación de tendencias y patrones, tanto en el tráfico de red como en los equipos que componen dicha red [5]. Por tanto, una monitorización precisa y eficiente

es vital para asegurar que la red opere de acuerdo con el comportamiento planificado y para solucionar posibles desviaciones en caso de que las hubiera. Todo ello ayuda a los administradores de red a determinar el rendimiento de la red y a mejorar su eficiencia. Según los autores de [6], existe una amplia gama de aplicaciones donde resulta de interés identificar los siguientes hechos: i) cuándo cambia la estructura de una red a lo largo del tiempo; ii) si alguna parte de la red es diferente del resto, en un modo significativo. La monitorización de red es el campo que se ocupa de identificar tales cambios. Por todo estas razones, actualmente, existe gran interés por parte de las compañías de telecomunicación en conocer el rendimiento de los sistemas de monitorización de red.

Varias aplicaciones pueden estar relacionadas con los sistemas de monitorización y análisis de tráfico; por ejemplo, sistemas de detección de intrusión [7], antivíruses de red [8] y monitorización de tráfico multimedia en tiempo real [9], entre otras.

2.1.1.1. La monitorización dentro de la gestión de red

Según [10], la monitorización puede ser considerada como parte de la gestión de red , concretamente, dentro de un ciclo de operaciones de gestión de red. Tal y como muestra la Figura 2.1, a medida que la red evoluciona en el tiempo, tres operaciones se suceden de forma secuencial: diseño, desarrollo y monitorización.

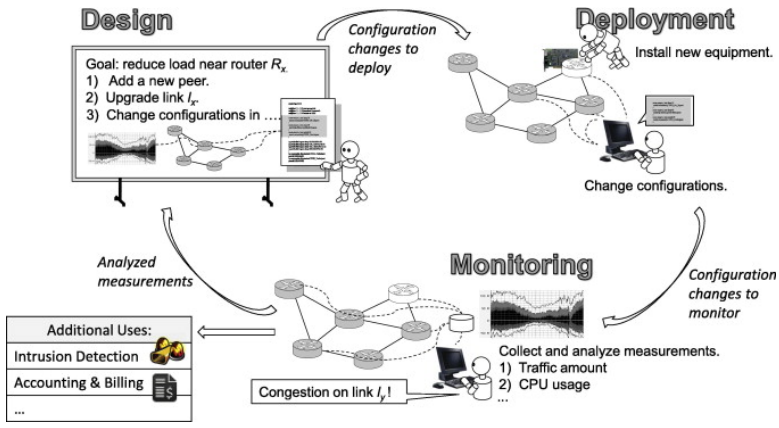


Figura 2.1: Operaciones de gestión de red. Fuente: [10].

Las operaciones de monitorización se encargan de recoger y analizar una serie de medidas que muestran el comportamiento actual del sistema de red. Por ejemplo, el número de paquetes capturados en diferentes puntos de observación, información específica de un equipo de red como puede ser la carga de CPU o el estado de las tablas de rutado de un router. La monitorización también ayuda en el seguimiento de la utilización de red por parte de los usuarios, para, por ejemplo, tareas de tarificación.

Las operaciones de diseño generan los cambios necesarios en la configuración y la infraestructura de red, para que se cumplan unas especificaciones. Si las operaciones de monitorización identifican un problema de rendimiento, este debe ser analizado minuciosamente y correlado con las configuraciones, para poder identificar las posibles causas. Las operaciones de diseño modificarán las configuraciones, en función de las causas detectadas.

Las operaciones de despliegue se encargan de implantar los cambios planificados, minimizando posibles perturbaciones del funcionamiento normal de la red. Tras hacer el despliegue, las operaciones de monitorización supervisan el comportamiento de la red con objeto de que se asegure que la red funciona según el comportamiento previsto.

2.1.1.2. Etapas de la monitorización

Las operaciones de monitorización se pueden clasificar según las diferentes funciones lógicas que llevan a cabo. Así, es posible distinguir cinco capas o etapas: captura, representación, informe, análisis y presentación.

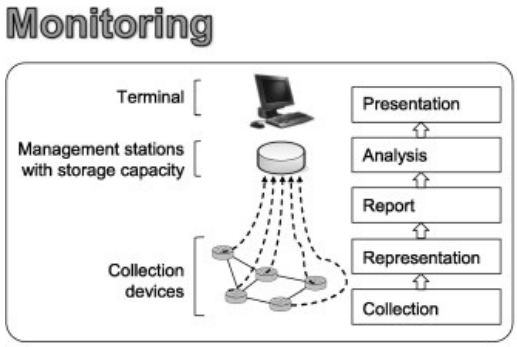


Figura 2.2: Capas de la monitorización de red. Fuente: [10].

La capa de captura recoge de la red los datos de medida en bruto . A continuación, estos datos son procesados en la capa de representación, para que sean puestos en un formato determinado. A menudo, se pretende que este formato sea válido para diferentes funciones de gestión. La capa de informe (*report*) transfiere los valores de medidas tomados desde un grupo de equipos distribuidos por la red a un conjunto más reducido de estaciones de gestión donde, habitualmente, se ubican funciones de nivel superior. La capa de análisis procesa los datos recibidos y extrae de ellos interpretaciones de nivel superior. Finalmente, la capa de presentación se ocupa de presentar los resultados extraídos del análisis a los operadores de red en diferentes formatos (tanto en modo gráfico como en modo texto).

Los sistemas de análisis de tráfico presentan diversos esquemas de funcionamiento o metodologías de análisis. Por otro lado, dentro de la etapa de captura, puede haber diversos esquemas de funcionamiento:

- Captura mediante monitorización activa. La denominación de activo para los sistemas de análisis se refiere a que los paquetes que se analizan han sido introducidos activamente en la red, bien por la arquitectura que realiza las medidas o bien por otros medios (inyectores). Existe el riesgo de que el tráfico inyectado, por su carácter intrusivo, pueda afectar al comportamiento natural de la red que se pretende analizar, ofreciendo resultados poco representativos. Su principal ventaja es que solo procesan las tramas introducidas deliberadamente para realizar la medición, por lo que controlan en todo momento el nivel de procesamiento al que son sometidos los elementos de la arquitectura de medida.

- Captura mediante monitorización pasiva. Los sistemas pasivos capturan y analizan el tráfico de la red sin introducir tráfico externo o adicional. De esta forma, los resultados obtenidos, considerando un sistema ideal sin pérdidas, reflejan exactamente lo que está ocurriendo en la red, sin fuentes externas de error.

- Captura mediante sistemas mixtos. Los sistemas mixtos combinan el análisis pasivo con la inserción de tráfico específico de medida. Dado que se realiza un análisis activo, se necesita una infraestructura con inyectores de tráfico y receptores.

2.1.2. Arquitectura de referencia de una sonda de análisis de tráfico de red

Después de ver las diferentes capas o etapas que pueden verse implicadas en un sistema genérico de monitorización de red, en este apartado, se describe el modelo de referencia concreto que se considera en este trabajo de Tesis. Este se centra en equipos que se comportan como sondas de análisis de tráfico de red.

En la Figura 2.3, se puede observar la estructura genérica de un Sistema de Análisis de Tráfico, o *Traffic Analysis System (TAS)*, de este tipo. Está compuesto por tres módulos: captura, filtrado y análisis. En el mejor de los casos, todos los paquetes que circulan por la red son capturados por la interfaz de red, que pertenece al módulo de captura. A continuación, se realiza un tratamiento básico con el que se filtran los paquetes, ya que no todos tienen por qué ser procesados. Finalmente, los paquetes que han superado el filtro son procesados en el módulo de análisis.

Tradicionalmente, las dos primeras etapas, la captura y filtrado previo de los paquetes, se realizan en el núcleo o *kernel* del sistema operativo de la máquina, mientras que un filtrado más avanzado y el análisis de los paquetes son tareas que se suelen construir en el nivel de usuario, al ser más fácil su desarrollo y portabilidad. El software de captura de paquetes se basa en las características proporcionadas por el sistema operativo. Este es el encargado de capturar los paquetes y transportarlos desde la interfaz de red hasta el área de memoria de usuario. En un sistema operativo convencional los mecanismos de captura y análisis de datos no suelen estar optimizados. Esto es lógico, ya que la mayoría de los sistemas operativos

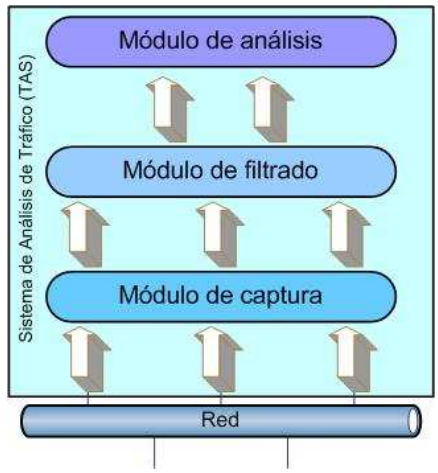


Figura 2.3: Arquitectura de referencia de una sonda de análisis de tráfico de red.

son de propósito general y no están específicamente diseñados para realizar estas tareas.

2.1.2.1. Módulo de captura

Las dificultades de los sistemas de análisis de tráfico para capturar todos los paquetes que circulan por la red se llevan intentando resolver desde los albores de las redes de datos. El módulo de captura es el encargado de extraer los paquetes de la red y ofrecérselos al *kernel* para su manipulación, haciendo uso de las tarjetas de red o *Network Interface Cards (NICs)*.

Las tarjetas de red convencionales pueden ofrecer buenos rendimientos aunque, sin optimizar sus drivers, son incapaces de capturar todos los paquetes a velocidades muy elevadas. Esto supone un problema de escalabilidad importante dado que existe una gran variedad de tarjetas diferentes con sus respectivos drivers que habría que modificar.

El paso de los paquetes desde la tarjeta de red hasta el nivel de usuario supone un coste computacional. Por esta razón, una de las líneas de investigación que más relevancia han tenido en este campo se ha centrado en el desarrollo de tarjetas de captura de datos con hardware especial [11]. Este hardware está específicamente diseñado para la captura de tráfico permitiendo alcanzar velocidades cada vez mayores. De ese modo, es posible reservar la capacidad de procesamiento para aplicar algoritmos complejos de tratamiento software a los paquetes. Por contra, uno de los inconvenientes de estas tarjetas es su elevado precio.

En muchos casos, donde las exigencias de captura no son tan extremas, es posible que una optimización del software de captura sea suficiente para analizar todo el tráfico de red. Esto permitiría utilizar plataformas hardware de propósito general a modo de sondas para el análisis de tráfico.

Otro aspecto importante relacionado con el módulo de captura es la sincronización de los paquetes. El incremento de las velocidades hace que el número de paquetes que circulan por la red cada segundo sea muy elevado obligando a que las marcas de tiempo deban ser mucho más precisas. A pesar de sus deficiencias, antes, el protocolo *Network Time Protocol (NTP)* era suficiente para tener un conjunto de ordenadores sincronizados. Si se deseaba más precisión se utilizaba el sistema *Global Positioning System (GPS)* que ofrecía 1 μs de precisión aproximadamente. Sin embargo, con el incremento del ancho de banda son necesarias nuevas técnicas para poder correlar información recogida por sensores no muy distantes, en los que las diferencias de tiempo son muy pequeñas.

Por otro lado, la llegada de las redes Gigabit exigió, a los sistemas operativos, una evolución en el diseño software de sus subsistemas de red. Se desarrollaron diferentes soluciones para los mecanismos de captura:

- Captura basada en interrupciones.
- Captura basada en *polling*.
- Captura basada en esquemas mixtos.

A continuación, se describen estos tres mecanismos.

Captura basada en interrupciones. Los primeros sistemas operativos utilizaban un mecanismo basado en interrupciones para planificar las operaciones de red. Es decir, cada vez que llegaba un paquete a la tarjeta de red, ésta generaba una interrupción y se ejecutaba la rutina de atención correspondiente a esa interrupción. En esa rutina, el *kernel* realizaba, sobre el paquete recién capturado, todas las operaciones de red que fueran necesarias. Una vez que la rutina había finalizado se continuaba con la tarea interrumpida. Este proceso se repetía con cada paquete que llegaba a la tarjeta.

Las ventajas de este mecanismo son la facilidad en su implementación y la baja latencia con la que los paquetes capturados se entregan al nivel de usuario. Sin embargo, tiene un inconveniente muy importante. Cuando la frecuencia de llegada de paquetes al sistema es alta, la mayor parte de los recursos de la CPU se consumen en las rutinas de atención a las interrupciones provocadas por la llegada de esos paquetes. El resto del tiempo disponible se repartirá entre las tareas activas del sistema. Si la tasa de llegada crece mucho, las interrupciones monopolizarán el uso de la CPU, impidiendo la ejecución de cualquier otra tarea y llegando a una situación de bloqueo, que en [12] se denominó *livelock*. De esta situación no se podrá salir hasta que disminuya la tasa de llegada de paquetes.

Captura basada en polling. Para evitar la degradación del rendimiento y las situaciones de livelock se pueden emplear mecanismos de *polling* [13], es decir, el sistema operativo pregunta periódicamente a la tarjeta si ha llegado algún paquete. En caso afirmativo, se procesarían todos los paquetes que hayan llegado hasta ese momento. De esta forma, aunque la

tasa de llegada sea muy elevada, el sistema no se bloquea, ya que es el propio *kernel* quien controla el consumo computacional del dispositivo de red y, por tanto, el número de paquetes que entran al sistema. Un ejemplo de aplicación del mecanismo de *polling* al sistema de captura de FreeBSD se presenta en [14].

Este mecanismo también tiene inconvenientes, ya que en situaciones en las que la tasa de llegada sea muy baja, se estará interrogando a la tarjeta de red a pesar de que no haya recogido ningún paquete, con el consumo que ello implica. Además, cuando la tasa de llegada es elevada, durante cada *polling* el sistema operativo captura varios paquetes consumiendo recursos por cada uno de ellos. En general, el sistema operativo no será capaz de determinar cuántos paquetes podrá analizar el sistema de captura durante ese *polling* y, dado que la tasa es alta, se podrían haber consumido recursos innecesarios al capturar paquetes que el sistema no va a ser capaz de procesar.

Captura basada en esquemas mixtos. Si bien el mecanismo de *polling* es la alternativa opuesta a los sistemas basados en interrupciones, sus inconvenientes lo hacen poco recomendable para su uso en sistemas de captura de tráfico. Los autores de [12] presentaron un detallado estudio que concluía que la incapacidad de los sistemas operativos para capturar todos los paquetes se debía al diseño de los mismos. Hasta ese momento, la captura de paquetes se realizaba mediante interrupciones hardware. Sin embargo, un mecanismo mixto capaz de evitar los *livelock* y a la vez reducir los consumos innecesarios debidos al *polling*, mejoraría el rendimiento del módulo de captura.

Para ello, Mogul definió la técnica denominada "coalescencia de interrupciones" en la que el *kernel* procesa varios paquetes por cada interrupción. Su funcionamiento es el siguiente:

- Cuando un paquete llega al sistema, se produce una interrupción hardware. Como siempre, tras la interrupción se llamará a la rutina de atención correspondiente, en la que se indicará al planificador que prepare una tarea de captura de paquetes, y se deshabilitarán las interrupciones. Cuando el sistema lo considere oportuno, ejecutará la tarea del *kernel* encargada de gestionar los paquetes recibidos y hacer un *polling* sobre la tarjeta de red.

- El número de paquetes capturado variará en función del tiempo que haya pasado desde que se produjo la interrupción hasta que el planificador haya podido ejecutar esta tarea. De esta forma, se capturan varios paquetes con el coste de una sola interrupción, mejorando el rendimiento. Una vez finalizada esta tarea, se vuelven a habilitar las interrupciones y el procedimiento se inicia de nuevo.

Este mecanismo es más eficiente, ya que cuando la frecuencia de llegadas sea baja, se comportará prácticamente como un sistema basado en interrupciones y cuando aumente se capturarán más paquetes por interrupción. Su mayor inconveniente es la latencia que introduce desde que el

paquete llega al sistema hasta que finalmente es atendido por la tarea del *kernel* encargada de su análisis. Este inconveniente y su influencia sobre los sistemas de análisis de tráfico ha sido estudiado en [15]; los autores concluyen que los efectos son más acentuados en los sistemas activos que en los pasivos.

2.1.2.2. Módulo de filtrado

A menos que se desee analizar todos los paquetes, el primer paso en el procesado debe ser el filtrado de los paquetes que no se desean. Hay que tener presente que los paquetes que no sean de utilidad deben ser eliminados cuanto antes para que no consuman recursos innecesariamente. Esto es lógico porque no tiene sentido capturar un paquete y llevarlo hasta la zona de memoria de usuario para que este lo deseche. Lo ideal sería que los paquetes fueran filtrados antes de llegar incluso a la tarjeta de red o que esta hiciera el filtrado [11].

Sin embargo, esto tiene ciertos inconvenientes, ya que el filtrado de paquetes puede ser dinámico, es decir, en algunas aplicaciones puede interesar filtrar unos paquetes u otros en función del comportamiento de la red y esto puede cambiar durante el funcionamiento del sistema de análisis de tráfico [16]. Por esta razón, los filtros a aplicar deben ser flexibles y su implementación hardware puede ser complicada en algunos casos.

Para afrontar esta tarea se han diseñado multitud de filtros, la mayoría de ellos en el *kernel* por las razones expuestas anteriormente. Los más famosos y más utilizados son los filtros *Berkeley Packet Filter (BPF)* [17].

Desde entonces, se han incorporado nuevos desarrollos y mejoras, dando lugar a los filtros *Fairly Fast Packet Filters (FFPF)* [18], *Swift* [19], *extended BPF (eBPF)* [20] o los presentados en [21] que se instalan sobre procesadores gráficos conocidos como *Graphical Processing Units (GPUs)*.

2.1.2.3. Módulo de análisis

Por último, se encuentra el módulo de procesamiento, generalmente a nivel de usuario. El software más utilizado para trasportar paquetes desde la tarjeta de red al nivel de usuario a través del sistema operativo es *libpcap* [22], aunque su rendimiento varía en función de la plataforma utilizada. Wireshark [23] también es otro ejemplo a señalar. El objetivo de estas utilidades es poner a disposición del usuario los paquetes extraídos directamente de la red.

El módulo de análisis puede realizar acciones muy diferentes en función del tipo de análisis de tráfico que se realice. Por ejemplo, en sistemas offline se encargarán de comprimir y almacenar la información mientras que en los híbridos se pueden actualizar algunos contadores de paquetes. Es en los sistemas online donde el procesado a nivel de usuario toma especial relevancia. El seguimiento de sesiones, el cálculo de las pérdidas, el retardo y el jitter, o la detección de intrusos y de virus precisan del mantenimiento de un estado en las conexiones y de la ejecución de algoritmos específicos, que se lleva a cabo en este nivel

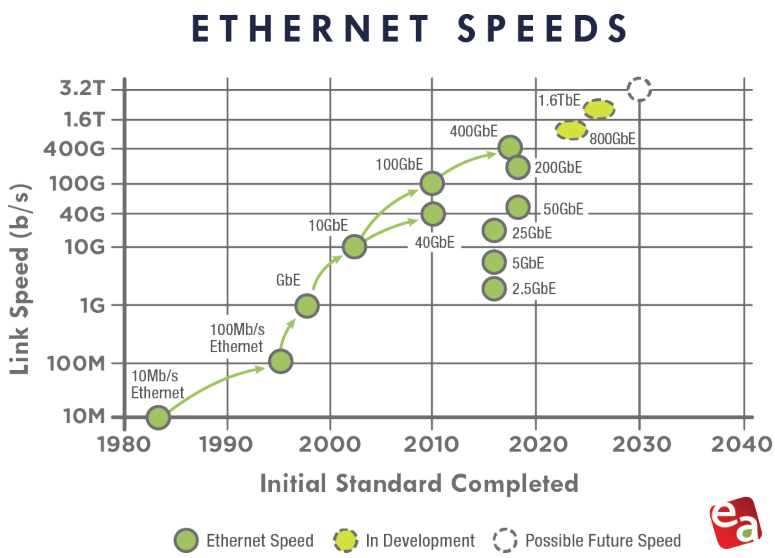


FIGURA 2.4: Ethernet Roadmap 2023. Evolución de velocidades de redes Ethernet. Fuente: [26]

2.1.3. Sistemas de análisis de tráfico de red con hardware de propósito general

A medida que aumenta el ancho de banda de las redes de datos, el volumen y la heterogeneidad del tráfico que estas manejan también es mayor. Esto hace que crezcan los requisitos funcionales de aplicaciones que capturen, procesen y/o almacenen el tráfico monitorizado. Trabajos como [24] ya describían, hace tres décadas, mecanismos relativamente complejos para capturar y marcar en tiempo los paquetes en una red Ethernet de 10 Mbps. Sin embargo, en el tiempo transcurrido desde entonces, las velocidades de las redes han incrementado considerablemente, dejando atrás, en algunos momentos, a las mejoras en velocidad de los procesadores, memorias y dispositivos de almacenamiento. La Figura 2.4 muestra la evolución de la velocidad de las redes Ethernet iniciándose con 10 Mbps, pasando por el Gigabit y llegando al multi-Gigabit de hoy en día. La asociación *Institute of Electrical and Electronics Engineers (IEEE)* [25] ya tiene estándares para 400 Gbps y, según el *roadmap* de 2023 de la Ethernet Alliance [26], se está trabajando en la definición de estándares para velocidades de 800 Gbps y 1,6 Tbps. Se dice que la era actual es la de "Terabit Ethernet (TbE)", aunque, por ahora, la gente se refiere a Terabit Ethernet para velocidades superiores a 100 Gbps.

A lo largo de los años, para poder adaptarse a este aumento de las tasas de red, el ámbito de la monitorización y análisis de tráfico en redes de datos ha dado lugar a numerosos proyectos e iniciativas de estudio. En muchos casos, la implementación se lleva a cabo sobre hardware de propósito general. Las propuestas pueden ser divididas en dos bloques:

- Propuestas para redes 1-10 Gigabit Ethernet.
- Propuestas para redes multi-Gigabit y 5G actuales.

A continuación, se describen algunas de ellas.

2.1.3.1. Propuestas para redes de 1-10 Gbps

Los primeros trabajos surgen porque se detectan importantes limitaciones en la captura y análisis de tráfico en redes 1 Gigabit Ethernet (1GbE) [27]. Por un lado, los subsistemas de red de sistemas operativos de propósito general (por ejemplo, Linux) no hacen un uso eficiente de los recursos hardware básicos de un equipo: el bus de memoria, la caché y, sobre todo, la capacidad computacional del procesador o procesadores [28]. Los recursos de procesamiento disponibles en los equipos se desperdician en tareas ineficientes de recopia de paquetes desde la tarjeta de red hasta la memoria de los procesos de aplicación, responsables en última instancia de procesarlos o almacenarlos, atravesando el *kernel* del sistema operativo, que a su vez introduce redundancia en el procesamiento. Todo esto conduce a que se produzcan pérdidas en el sistema y se invaliden los resultados de ciertos tipos de análisis. Las soluciones generadas para resolver esta problemática son, fundamentalmente, propuestas centradas en software instalado sobre hardware de propósito general (o *commodity hardware*). Su objetivo principal es mejorar el funcionamiento del subsistema de red y, al mismo tiempo, hacerlo de una manera efectiva desde el punto de vista de coste. Por ello, en terminología inglesa, se dice que se utiliza hardware y software *commodity-off-the-shelf* (COTS).

Es preciso indicar que también hay algunas propuestas enfocadas hacia el desarrollo de hardware específico: por ejemplo, i) el uso de tarjetas de captura de red de altas prestaciones [11]; ii) o de *Network Processors (NP)* [29] para el análisis de tráfico en tiempo real [30]; iii) o soluciones basadas en *Field-Programmable Gate Arrays (FPGA)* [31]. No obstante, son menos numerosas que las de software.

La primera propuesta software destacable es de Mogul y Ramakrishnan [12], que identificó los problemas de rendimiento más importantes de sistemas de captura basados en interrupciones. Además, realizó propuestas que aún están en vigor: limitar las interrupciones; utilizar el esquemas híbrido de captura denominado coalescencia de interrupciones para eliminar el problema de *livelock*; revisar los sistemas de planificación de tareas para ajustar tiempos de procesado frente a los de captura; optimizar agresivamente el camino de procesamiento de los paquetes en el sistema; o dimensionar adecuadamente los bufers.

A raíz del mencionado trabajo de Mogul y Ramakrishnan, surgen ideas para nuevas arquitecturas que constituyen una **primera etapa de propuestas**. Cada una de ellas sugiere alguna mejora de rendimiento, tal y como se describe a continuación:

- En primer lugar, destacan las numerosas aportaciones de Deri, tanto en esta primera etapa como en la siguiente (tal y como se podrá

apreciar más adelante). Así, se presenta el diseño y la implementación software de una sonda denominada nProbe [32] que captura paquetes sobre una red Gigabit Ethernet y los procesa como flujos Netflow [33]. Posteriormente, el resultado es enviado a otro software denominado ntop [34] que se encuentra en una entidad de nivel superior del sistema de monitorización de red. El mismo autor presenta su mecanismo de captura denominado PF_RING [35]. Este es una modificación del núcleo o *kernel* de Linux que mejora la transferencia de paquetes desde la NIC hacia el espacio de usuario con técnicas de mapeo de memoria. De esta forma, se evita alguna tarea de copia que puede ser útil en un sistema operativo de propósito general, pero que es innecesaria en sistemas para la captura pasiva de paquetes. PF_RING se instala sobre el sistema operativo Linux convencional con su correspondiente driver.

- Trabajos desarrollados por Hurwitz y Feng [36][37] establecen que la limitación más importante en el análisis la imponen los elementos de procesamiento. Los problemas surgen principalmente cuando la tasa de datos es alta y esto conlleva niveles de carga de CPU altos y pérdida de paquetes.

- En [38] se presenta una arquitectura de un sistema distribuido de monitorización de red a 10 Gbps que divide sus tareas en cinco fases: captura de paquetes, generación de flujos, almacenamiento de flujos, análisis y presentación.

- En [39] se presenta el diseño preliminar de una infraestructura distribuida de monitorización pasiva llamada *Distributed Passive Monitoring Infrastructure (DPMI)*. Consta de tres elementos: los nodos de medida, los consumidores (destinos para las medidas) y las áreas de medida (que controlan los nodos). Mediante una interfaz web, los usuarios pueden configurar las medidas que desean realizar en base a filtros relativamente complejos.

- También es de esta etapa la sonda denominada Ksensor [40], desarrollada por el grupo NQaS de la UPV/EHU. Se trata de una arquitectura que captura y analiza paquetes a nivel de *kernel*, con sistema operativo Linux y hardware de propósito general.

- Braun *et al.* [41] evalúan y comparan diferentes soluciones de captura de paquetes sobre Linux y FreeBSD de esta primera etapa. Su conclusión es que PF_RING es la que mejor rendimiento ofrece.

- Asimismo, dan comienzo iniciativas europeas reseñables como el proyecto BISMark [42], TRAMMS [43], GEANT2 y la *COST Action IC0703 Data Traffic Monitoring and Analysis: theory, techniques, tools and applications for the future networks* [44].

Posteriormente, los avances tecnológicos de los equipos hardware de propósito general posibilitaron la captura y procesado de tráfico en redes 10 GbE, reduciendo las pérdidas de paquetes [45]. Ejemplos de estos

avances son el uso extendido de CPUs con varios núcleos (o *multi-core*), la tecnología denominada *Receive Side Scaling (RSS)* [46] que distribuye la carga de tráfico entre diferentes colas y optimiza el uso de la memoria caché, la aparición de tarjetas de red 10GbE que disponen de filtros hardware para distribuir el tráfico entre diferentes núcleos y la arquitectura de memoria denominada *Non-Uniform Memory Access (NUMA)* que asigna la memoria de una forma más eficiente tanto en entornos multiprocesador como en *multi-core*.

Todo ello confirma que los sistemas basados en *commodity hardware* son muy atractivos para la monitorización de tráfico de redes de alta velocidad, puesto que su rendimiento puede ser comparado con el de hardware especializado como FPGAs (por ejemplo, NetFPGA), tarjetas de red, *network processors*, o soluciones comerciales proporcionadas por fabricantes de routers. Además, puede adquirirse por un precio significativamente menor; por tanto, proporcionan soluciones accesibles desde el punto de vista de coste. Además, dado que, habitualmente, la funcionalidad de monitorización es desarrollada a nivel de usuario, las soluciones basadas en *commodity hardware* son altamente flexibles, lo que permite diseñar sistemas escalables.

Con esta nueva situación, surge otro conjunto de ideas que se engloban en una **segunda etapa de propuestas**, tales como:

- PF_RING ZC (Zero Copy) [34] es la más reciente de las diferentes versiones de PF_RING de Deri. Consigue procesar paquetes en redes de hasta 10 Gbps. Su versión previa, llamada PF_RING DNA/LibZero, ya permitía crear caminos paralelos desde las colas de recepción hardware a los procesos de usuario, esto es, asignar un núcleo de CPU a cada cola de recepción. Este motor de captura implementa preasignación y reutilización de memoria en todos sus procesos. La memoria puede ser asignada observando los nodos NUMA. Además, PF_RING ZC implementa *full zero-copy* mapeando memoria de espacio de usuario en la zona de memoria del driver habilitada para acceso directo a memoria o *Direct Memory Access (DMA)*. Así, las aplicaciones de usuario acceden directamente a los registros de la tarjeta y a los datos, evitando la intermediación del búfer de paquetes de *kernel* y reduciendo el número de copias. Hoy en día, PF_RING ZC está preparado para trabajar en entornos de máquinas virtuales *Kernel-based Virtual Machine (KVM)*. Además dispone de una librería para Snort (uno de los más conocidos sistemas de prevención y detección de intrusión) denominada *Snort Data Acquisition*. También hay una propuesta denominada vPF_RING para mejorar el rendimiento de captura sobre máquinas virtuales [47].

- PacketShader [48] es un router software capaz de trabajar a tasas multi-10Gbps. Sus autores también desarrollaron su propio motor de captura. Por ello, también pueden analizarse sus tareas relacionadas con la captura y procesado de paquetes. Se caracteriza por implementar mapeado de memoria. Así, se permite que los usuarios

acceden a los bufers de *kernel* y se evitan copias innecesarias. Asimismo proporciona paralelismo en el procesamiento de paquetes a nivel de usuario, por lo que se equilibra la carga de CPU y da escalabilidad en el número de núcleos y colas. Por último, indicar que utiliza técnicas de *batching* a nivel de usuario para reducir la sobrecarga de procesamiento por paquete.

- La arquitectura Netmap [49], disponible para FreeBSD y Linux, comparte muchas de las características de PacketShader. Aplica preasignación de memoria durante la fase de inicialización, bufers de tamaño fijo (también de 2048 bytes), procesamiento *batch* y caminos directos paralelos. También implementa técnicas de mapeado de memoria que permiten que las aplicaciones de usuario accedan a los búferes de paquetes de *kernel* (el acceso directo a NIC está protegido). Utiliza una representación de metadatos simple y optimizada.

- PFQ [50] Este mecanismo permite capturar paquetes desde una aplicación de usuario con un grado de paralelismo ajustable. El enfoque de PFQ es distinto al de PF_RING, PacketShader y Netmap. En lugar de llevar a cabo importantes modificaciones en el driver con objeto de evitar el esquema de interrupciones de NAPI o mapear la memoria habilitada para DMA y los búferes de paquetes de *kernel* a espacio de usuario, PFQ propone una arquitectura general donde se pueden utilizar tanto drivers modificados como originales. PFQ implementa una nueva capa, denominada *Packet Steering Block*, entre el nivel de usuario y las colas *batching queues*. Esta capa distribuye el tráfico entre los diferentes sockets de recepción.

- HPCAP [51] es un sistema que permite la captura de tráfico a 10 Gbps con marcado de tiempo preciso. Además está optimizado para permitir el almacenamiento de ese tráfico en disco para su posterior análisis. El sistema es una modificación del driver original de Intel para tarjetas 10 GbE. La comunicación con las aplicaciones cliente de HPCAP se realiza a través de descriptores de fichero generados para cada interfaz. Dispone de mecanismos para evitar copias adicionales que afecten negativamente al rendimiento.

- DPDK [52] Desarrollado por, entre otros, Intel, *Data Plane Development Kit (DPDK)* es un conjunto de librerías y drivers para el envío y procesado de paquetes a alta velocidad. Soporta tarjetas de múltiples fabricantes.

- OpenOnLoad [53] es un motor de captura que se instala sobre sistema operativo BSD y opera a a nivel de usuario.

Por último, [54] presenta una comparativa entre varios motores de captura de esta segunda etapa (PF_RING ZC, DPDK y Netmap). Su conclusión es que todas estas variantes aplicadas al sistema operativo clásico aportan ventajas. Sin embargo, no destaca a ninguna de ellas por encima de las demás.

2.1.3.2. Propuestas para redes multi-Gigabit y 5G actuales

Con la llegada de las redes 5G y la propuesta de servicios virtualizados sobre *datacenters* en la nube, se crean unos entornos de operación complejos y muy dinámicos, donde las técnicas de monitorización tradicionales ya no son tan válidas. En este contexto, surgen las redes definidas por software, o SDN, cuya principal característica es la abstracción del plano de control sobre el plano de datos. Es posible monitorizar la red a través de la arquitectura SDN[55], teniendo la opción de realizarlo desde cualquiera de los dos planos. Si se hace desde el plano de datos, se denomina monitorización de red en banda y, si se hace desde el plano de control, monitorización de red fuera de banda. Los dispositivos SDN permiten registrar y transmitir los datos de monitorización de una forma más eficiente, flexible y, al mismo tiempo, sencilla que los agentes utilizados en los métodos tradicionales. Esto se puede ver en trabajos como [56] y [57]. El primero propone un sistema de monitorización de tráfico de red integrado con SDN y NFV, lo compara con el de una red tradicional y concluye que obtiene un rendimiento similar, pero con un menor coste. El segundo propone integrar el software de monitorización en el plano de gestión de la arquitectura SDN interactuando con los dispositivos de red que se encuentran en el plano de datos.

Otra forma de abordar la monitorización de servicios *cloud* es complementar las sondas tradicionales con nuevas herramientas que hagan el seguimiento de los indicadores de rendimiento. En [58], esto se logra distribuyendo unos agentes software ligeros que monitorizan servidores y dispositivos de red tanto físicos como virtuales. Cada uno de estos agentes recoge estadísticas de los elementos monitorizados y envía sus datos a un repositorio central provisto de herramientas de análisis *big data*. Aplicada la lógica correspondiente, el sistema central genera alertas, diagnósticos y visualiza la información de modo comprensible para los operadores de red.

Los trabajos [59] y [60] introducen el concepto de sonda virtual de red o *Virtual Network Probe (VNP)* . Puede definirse una VNP como una *Virtual Network Function (VNF)* que monitoriza el tráfico de elementos de red que pueden ser tanto físicos como virtuales. Una VNP representa un caso de que se alinea con el concepto de monitorización como un servicio o *Monitoring as a Service* (MaaS) [61]. En su propuesta para capturar datos de redes de 40 Gbps con estas VNPs, utilizan tarjetas de red comerciales COTS con el driver HPCAP40vf (evolución de la propuesta HPCAP expuesta anteriormente).

nTap [34] es otro ejemplo de sonda software virtual que puede ser usada en entornos *físicos*, virtuales y *cloud* para capturar paquetes y enviar datos al punto central de observación. nTap puede ser instalado mediante contenedores, máquinas virtuales y Kubernetes [62].

Es de interés mencionar los trabajos [64][63] de Pérez *et al.* que están alineados con la investigación que se presenta en esta tesis ya que muestran cómo sería el diseño de una sonda de monitorización de tráfico para redes 5G. Estos trabajos se enmarcan en un proyecto europeo denominado *5G EVE* y referencian trabajos realizados en otros proyectos como *5GROWTH* y *5G-TRANSFORMER* donde se han propuesto sondas de monitorización

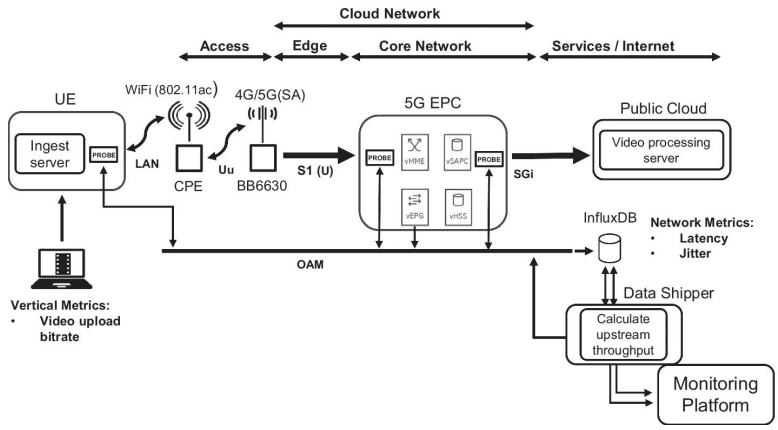


FIGURA 2.5: Ejemplo de monitorización de red 5G. Fuente: [63].

de tráfico similares. En este contexto, es de destacar también los esfuerzos de 3GPP en la estandarización de *Network Data Analytics Function (NWDAF)* y *Management Data Analytics Function (MDAF)* para redes 5G. Estas funciones definidas por los estándares [65][66] de 3GPP permite homogeneizar la forma de capturar datos y gestionarlos en los elementos de la red que precisan disponer de esa información y que pueden comprometer en alguna medida el rendimiento obtenido por las sondas de tráfico. Volviendo a la propuesta de Pérez, esta presenta un ejemplo de monitorización de un servicio de distribución de vídeo alojado en una nube pública (o *public cloud*). La conexión entre el equipamiento de usuario y el servicio es a través de una red 4G/5G. Para tomar medidas de ciertas métricas de rendimiento, tales como la latencia o el *jitter*, se despliega un conjunto de sondas en puntos clave de la red. La Figura 2.5 muestra la arquitectura de este caso donde se puede ver que existen sondas software (*probes*) instaladas en el equipamiento de usuario (o *User Equipment-UE*) y en la infraestructura de red 5G (o *Evolved Packet Core-5G EPC*). Además de los elementos monitorizados en este ejemplo, también son posibles otros como antenas de radio, funciones de red físicas o *Physical Network Functions (PNF)*, VNF u otras herramientas de monitorización integrables con esta.

Para terminar con esta revisión, indicar que, dentro de los equipos de monitorización de datos basados en *commodity hardware*, los sistemas basados en Linux han sido ampliamente utilizados [67] [68].

2.2. Revisión de modelos matemáticos aplicados a sistemas de análisis de tráfico

Al margen de las diferentes propuestas de arquitecturas de sistemas de captura y análisis vistas anteriormente, también existen estudios analíticos orientados hacia la evaluación de rendimiento de dichos sistemas computacionales. Algunos de ellos son basados en teoría de colas y tanto su análisis como su adaptación y aplicación tienen interés en las labores de investigación de esta tesis.

Los modelos matemáticos deben ser construidos en base a un sólido conocimiento del sistema real. Tal y como se recomienda en el manual de ingeniería de teletráfico [69], el proceso debe tener necesariamente un carácter iterativo. Después de aplicar el modelo, las características y resultados derivados del mismo serán comparados con datos medidos en el sistema real (si este existe). Si unos y otros no tienen concordancia, deberá llevarse a cabo una nueva iteración del proceso conjunto de modelado y validación. En definitiva, tal y como se indica en [70], es fundamental el uso de una metodología válida para llevar a cabo el estudio de evaluación de rendimiento, poniendo especial énfasis en la necesidad de utilizar el método científico; en concreto, Le Boudec propone una metodología válida para cualquier estudio de evaluación de rendimiento.

Los modelos basados en teoría de colas han sido ampliamente empleados para el modelado, análisis y predicción de rendimiento de varios sistemas de comunicación y redes de ordenadores [71]. Dichos modelos permiten representar la competencia de un conjunto de clientes por conseguir uno o varios recursos. Los servidores de las colas representan los recursos; los clientes, que ocupan las colas y luego obtienen el servicio, representan los trabajos, tareas, paquetes o mensajes que son la carga de trabajo del sistema. Los análisis de rendimiento de sistemas basados en teoría de colas evalúan un conjunto de parámetros de interés o índices de rendimiento como son el throughput del sistema, la utilización del recurso, el tiempo medio de respuesta de cliente o el retardo. Normalmente, el análisis de estos sistemas se basa en la definición del proceso estocástico subyacente.

En general, existe gran cantidad de trabajos que modelan redes de telecomunicaciones mediante teoría de colas, desde sistemas muy antiguos hasta los más recientes [72]. Cabe mencionar algunos de ellos, por ejemplo, [73] donde se construye un modelo analítico para predecir ciertas características de tráfico, indicadores de calidad de servicio o QoS y el rendimiento global de la red. En [74], se propone un modelo conceptual de un sistema de telecomunicación global mediante un sistema de colas; este modelo analítico permite resolver aspectos de dimensionamiento de red y QoS. Otros trabajos más recientes están relacionados con redes 5G; entre estos últimos, [75] y [76] son dos ejemplos de modelos analíticos aplicados a arquitecturas SDN de redes 5G.

Si se pone el foco en los modelos sobre sistemas de captura y análisis de tráfico de red, destaca la tarea realizada por el grupo investigador

encabezado por Salah. En sus primeros trabajos, tienen varias propuestas de modelado analítico sobre esquemas de recepción de paquetes basados en la gestión de interrupciones [77][78][79]. Modelan los siguiente casos: el ideal para la captura de paquetes con un tamaño de búfer limitado; el correspondiente al modo de interrupción tradicional en el que cada paquete entrante causa una interrupción; el del comportamiento con coalescencia de interrupciones; y también un modo de funcionamiento que habilita y deshabilita interrupciones. Sus modelos analíticos están basados principalmente en colas finitas M/M/1/B y cadenas de Markov que representan el comportamiento de cada caso. Generalmente validan sus resultados analíticos con simulación, aunque también disponen de alguna comparación con datos de sistemas reales.

Siguiendo con el mismo autor, sus aportaciones evolucionan de sus primeros modelos hacia distintas aplicaciones también relacionadas con la captura y análisis de tráfico. Así, en [80] se realiza una comparativa del rendimiento del subsistema de red de los sistemas operativos Linux y Windows, donde las métricas comparadas son el throughput, la pérdida de paquetes, el retardo y la disponibilidad de CPU. Por otro lado, en [81] se caracteriza el sistema de detección de intrusión Snort ejecutado sobre Linux con un modelo analítico de colas. Dicha representación del comportamiento del *kernel* del sistema operativo y del Snort identifica factores de impacto que son claves en el rendimiento del Snort. Este mismo equipo investigador ha modelado y analizado el rendimiento de routers software basados en arquitectura PC [82]. Este modelo, basado en sistemas de colas finitas ha sido verificado y validado posteriormente mediante simulación, así como a través de la obtención de medidas experimentales sobre un PC-router con dos interfaces. Por último, en [83] modelan el rendimiento de un firewall de red con una serie reglas mediante un modelo analítico de colas basado en cadenas de Markov. Para terminar con Salah, indicar que posteriores trabajos suyos siguen utilizando técnicas de modelado similares aplicadas sobre máquinas virtuales y *cloud computing* [84][85][86].

Otro trabajo con un enfoque parecido al anterior, aunque con notables diferencias, es el desarrollado por Wu [87]. En él también se propone un modelo matemático para caracterizar y analizar un proceso de recepción de paquetes, concretamente el del subsistema de red de Linux. El modelo tiene una parte basada en el algoritmo *token bucket*, que está relacionada con la tarjeta de red y el driver del dispositivo de red, y otra, que modela el resto de la recepción de paquetes, en teoría de colas.

Por otro lado, los autores de [88] modelan una arquitectura de un switch OpenFlow mediante un sistema de colas compuesto una cola M/GI/1 para el tráfico saliente y una cola finita M/M/1/B que representa al controlador OpenFlow. El modelo analítico es validado mediante simulación en OMNeT++.

A su vez, existen varios trabajos relacionados con modelado de firewalls. En [89], se propone un modelo analítico para analizar firewalls utilizando distribuciones exponenciales e hiper-Erlang. El autor de [90] y [91] también evalúa el rendimiento de firewalls de la capa de aplicación mediante teoría de colas.

Hay varias investigaciones para medir el consumo energético de los motores de procesado de paquetes [92][93]. Entre ellos, cabe mencionar [92]. Este propone una cola $M^x/D/1$ con un periodo de *setup* e incorpora la resolución de un problema de optimización que estudia el balance entre el consumo energético y el cumplimiento de los índices de rendimiento de red

Por último, otro trabajo interesante es el de [94] que formula una cola $M^x/G^B/1/K$ para analizar el rendimiento de las principales técnicas adoptadas por tres de los principales motores de captura de paquetes (Netmap, Intel DPDK y PF_RING ZC) en redes de alta velocidad

2.3. Servicios de redes 5G susceptibles de ser modelados con teoría de colas

El contexto de investigación que inició esta Tesis Doctoral corresponde a la línea de investigación del grupo NQaS referente a "Sondas de tráfico en redes de alta capacidad". Para ello, se ha investigado en el desarrollo de modelos matemáticos que ayuden en el estudio del rendimiento de sondas propias diseñadas para la captura y análisis de tráfico en redes fijas de alta capacidad (Gigabit y multi-Gigabit Ethernet).

A lo largo de estos años, el grupo NQaS ha desarrollado una fuerte investigación en la línea denominada "Despliegue y orquestación inteligente de servicios en entornos 5G, B5G y 6G". En este nuevo contexto, surge interés en integrar el conocimiento adquirido en el modelado de sistemas basados en teoría de colas en la nueva línea de investigación.

Se plantean, en este tipo de redes B5G, dos líneas de estudio donde la teoría de colas puede ser aplicable:

1. Modelado de una sonda B5G de captura y análisis de tráfico. Tecnologías como NFV y SDN permiten realizar las funciones de captura y análisis de tráfico de una forma más flexible en la infraestructura de red.

2. Modelado de sistemas de provisión de servicio extremo a extremo en redes B5G, para estudiar la distribución de los servicios basados en entidades virtualizadas (VNF) que pueden ser desplegadas en diferentes partes de la red, aprovechando capacidades de estas redes incluso en el extremo de la misma (*Multi-Access Edge Computing-MEC*).

Dada la experiencia del grupo NQaS en la segunda línea de estudio, es de especial interés realizar investigación sobre los servicios de Misión Crítica (MC). En particular, se plantea como tercera línea de estudio la tecnología *Mission Critical Push To Talk (MCPTT)*.

Seguidamente se plantean las directrices de cada una de esas líneas de estudio, prestando especial atención a la tercera sobre modelado de servicios MCPTT, ya que en esta tesis se plantea un caso de estudio específico sobre esa tecnología que, al funcionar sobre una arquitectura B5G, es también un caso de ejemplo de la segunda línea de estudio.

2.3.1. Modelado de una sonda B5G de captura y análisis de tráfico

La arquitectura de un nodo de la red de acceso radio (eNodeB o gNodeB), se asemeja mucho a una sonda de tráfico tradicional. Es más, mucho equipamiento de estas redes se puede considerar como sondas de tráfico. En general, la provisión de su servicio supone capturar tráfico y realizar una función específica de análisis del mismo que, en función de la disponibilidad de recursos y de cómo sean las demandas de servicio, pueden comprometer los indicadores de calidad de rendimiento del sistema, generalmente medidos a través de sus indicadores de rendimiento clave o *Key Performance Indicator (KPI)*.

En la arquitectura *Long Term Evolution (LTE)* se tienen dos partes bien diferenciadas: la red de acceso radio o *Radio Access Network (RAN)* y un núcleo de red basado en la arquitectura TCP/IP denominado *Evolved Packet Core (EPC)*. En la parte RAN están los nodos de acceso radio (eNodeB). En la parte EPC se sitúan una serie de nodos que permiten desarrollar las funciones correspondientes a un núcleo de red de una red móvil celular: la entidad de gestión de movilidad o *Mobility Management Entity (MME)*, el sistema servidor de abonados domésticos o *Home Subscriber Server (HSS)*, la pasarela de servicio o *Serving Gateway (S-GW)*, así como la pasarela de red de paquetes de datos o *Packet Data Network Gateway (PDN-GW)* que actúa como punto de interconexión con redes IP. Todos estos elementos son susceptibles de modelar como si fueran una sonda de tráfico clásica.

El 3GPP identificó en la Release 14 la necesidad de separar el plano de control y el plano de usuario definiendo la arquitectura *Control and User Plane Separation (CUPS)*. Esto es debido al crecimiento del tráfico de datos en multitud de dispositivos y, con tráfico multimedia, a la necesidad de reducir latencias y cumplir con KPIs críticos y a las ventajas de escalabilidad de la infraestructura de red al separar los planos de control y usuario. Esta separación del plano de usuario permite utilizar tecnologías como redes SDN, para distribuir los datos de una forma más eficiente. Aparece una entidad nueva como es la función de detección de tráfico o *Traffic Detection Function (TDF)* que se encarga de detectar y reportar la descripción de flujos de datos identificados en los servicios y aplicaciones. Esta labor de monitorización es típica en las sondas de tráfico que son objeto de esta tesis.

En la arquitectura 5G ya no es tan fácil identificar las entidades o nodos conectados entre sí a través de interfaces punto a punto como se propone en las arquitecturas anteriores. Se define una arquitectura orientada a servicio o *Service-Based Architecture (SBA)* donde se habla de funciones a realizar que pueden estar disponibles para el resto de la arquitectura. Se basa en el paradigma de la virtualización de funciones. La parte de RAN de 5G, en esencia, es parecida a la anterior de 4G-LTE y lo que cambia es la parte de señalización del *Core* que es distinta a la planteada en EPC de LTE.

Algunas de las funciones definidas en el *Core* de 5G son:

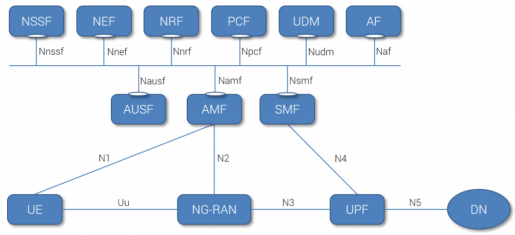


FIGURA 2.6: Funciones en la arquitectura 5G. Fuente: [95].

- Gestión del acceso y movilidad o *Access and Mobility Management Function (AMF)*. Gestiona la señalización de los dispositivos para el registro, autenticación y gestión de la movilidad.
- Gestión de la sesión o *Session Management Function (SMF)*. Administra las sesiones de los usuarios y direccionamiento IP.
- Plano de usuario o *User Plane Function (UPF)*. Gestiona el tráfico de datos del usuario con las redes externas IP.
- Gestión de datos unificada o *Unified Data Management (UDM)*. Gestiona las credenciales de los suscriptores móviles y dispositivos.
- Servicio de autenticación o *Authentification Server Function (AUSF)*. Se encarga de la autenticación de dispositivos según sus credenciales.
- Repositorio de funciones de red o *Network Repository Function (NRF)*. Realiza el registro de las funciones y servicios en la red.

El estándar de 3GPP no entra en cómo se deben implementar estas funciones ni sobre qué arquitectura física. Al tratarse de funciones que se pueden virtualizar, se pueden utilizar tecnologías como NFV para su diseño, implementación y despliegue.

Tradicionalmente, los operadores de las redes móviles célulares instalan equipos físicos específicos que realizan determinadas funcionalidades en la arquitectura de red y que, difícilmente, pueden ser reprogramados para proveer otros servicios distintos a los que han sido diseñados. Actualmente, con la virtualización de esos servicios, el equipamiento físico que precisan los operadores puede ser más flexible y se puede reutilizar mucho más fácilmente incluso con los cambios tan frecuentes en la estandarización. Una sonda para la gestión de tráfico en estas nuevas redes debe contemplar el paradigma de la virtualización y la posibilidad de distribución de los servicios de computación de una forma más flexible.

Uno de los problemas más importantes en las sondas de análisis de tráfico es la pérdida de paquetes. Si se pierde un paquete antes de alcanzar su destino, se están desperdiciando muchos de los recursos usados para la transmisión. Se pueden producir también efectos de congestión que producen retardos no deseados, por lo cual es importante disponer de algoritmos adecuados de gestión de colas. Las técnicas basadas en teoría de colas son

de interés para proponer diferentes algoritmos de gestión de colas en la interfaz de radio y controlar parámetros de rendimiento de interés como son la pérdida de paquetes, el throughput y el retardo extremo a extremo.

En los estándares de 3GPP sobre redes 5G, se han contemplado tres tipos de servicio: (a) banda ancha móvil mejorada o *enhanced Mobile Broadband (eMBB)*, comunicaciones masivas entre máquinas o *massive Machine-Type Communications (mMTC)* y comunicaciones ultrafiables de baja latencia o *ultra-High Reliability & Low Latency Communications (uRLLC)*. Así que, es necesario diseñar el equipamiento de la red 5G para soportar estos servicios y para garantizar el cumplimiento de los indicadores KPI que pueden exigir por cada una de las aplicaciones que se puedan demandar. Incluso en 5G, aparece un concepto como es segmentación de red (*network slicing*) que permite ofrecer múltiples redes lógicas aisladas sobre una red física única. Esto implica tener que armonizar los recursos disponibles para que cada segmento de red provea los servicios adecuados a cada tipo de usuario según sus compromisos de servicio.

El problema principal que se ataca en esta línea es el estudio del compromiso que existe entre la capacidad computacional de las sondas de análisis de tráfico en función de su carga de análisis y el incremento del caudal de tráfico a atender. En situaciones de saturación de recursos por aumento del caudal de tráfico o por las demandas asociadas de recursos computacionales no disponibles se pueden producir pérdidas de paquetes que comprometen los indicadores de rendimiento del sistema. Igualmente interesa estudiar la evolución de los indicadores KPI en función de las estrategias de gestión de los recursos disponibles. Para todo ello, es de interés el desarrollo de modelos matemáticos basados en teoría de colas.

2.3.2. Modelado de sistemas de provisión de servicio extremo a extremo en redes B5G

Ampliando el área de estudio más allá de los nodos de acceso a la red 5G y contemplando la cadena completa de provisión de un servicio extremo a extremo, en este tipo de redes, existen necesidades de crear modelos matemáticos que ayuden en la planificación adecuada de los recursos de la red 5G, donde la teoría de colas tiene su aplicabilidad.

La teoría de colas es una buena herramienta para modelizar redes 5G por las razones que se apuntan a continuación:

- Mucho del tráfico en redes 5G, como el control de dispositivos en la IoT y las comunicaciones entre máquinas (*Machine-type communications-MTC*), se basan en respuestas cortas tipo REST que son más fácilmente modelables con teoría de colas.

- En la provisión de un servicio, en estas redes, se producen varias etapas de procesamiento que disparan a su vez múltiples peticiones que pueden ser representadas con topologías de redes de colas.

- La virtualización de servicios y tecnologías como la computación en el extremo de la red móvil, *Multi-access Edge Computing (MEC)*,

pueden modelarse como tasas de servicio de colas asociadas a la capacidad computacional de esos procesos.

Tecnologías como NFV, que fueron estandarizadas por ETSI en 2012 [96], son claves para el desarrollo de las redes 5G. La arquitectura NFV permite sustituir el enfoque tradicional de equipamiento físico específicamente diseñado para desarrollar un servicio, por módulos software que pueden gestionar funciones virtuales de red (VNF). Esto proporciona a cada función de la red 5G la modularización y el aislamiento necesario para operar de forma independiente en un entorno virtual de propósito general.

De esta forma, se puede definir un servicio de red virtualizado combinando VNF individuales y se puede representar su composición en un gráfico donde cada función está identificada, tiene un orden y está localizada en una posición. Este tipo de gráficos son muy útiles para ordenar la investigación y se plantean como problemas clave: (a) la optimización en la ubicación de recursos, (b) la definición de la cadena de provisión de VNF y (c) la planificación en la ejecución de esas VNF. La teoría de colas tiene una aplicabilidad clara en la investigación de estos problemas.

Otra de las tecnologías clave en el desarrollo de las redes 5G, y que se puede considerar como complementaria a NFV, son las redes SDN, que permite una gestión más sencilla y ágil del tráfico de la red al utilizar controladores basados en software e interfaces de programación de aplicaciones (API) para dirigir los flujos de datos, de forma inteligente y programable, hacia la infraestructura de red subyacente.

Claramente, la virtualización de las funciones de red que propone NFV, y de los equipos, controladores y gestión de flujos que propone SDN, supone la necesidad de disponer de mayor cantidad de recursos computacionales en la infraestructura de la red 5G. Surge así el concepto de computación en la nube (*Cloud Computing*), que permite a la infraestructura 5G disponer de recursos computacionales en diferentes partes de la red: (a) en los centros proveedores de servicios de aplicación (*Application Services-*), (b) cerca del extremo de la red (*Cloud Enabled RAN-CE-RAN*) e incluso (c) en los dispositivos de los usuarios (*Mobile Cloud Computing-MCC*).

Siguiendo esta estrategia, ETSI ha hecho un gran esfuerzo por estandarizar la arquitectura ETSI MEC [97] que desarrolla la tecnología de computación en el extremo de la red móvil o *MEC*. Esta tecnología permite ejecutar servicios cerca de los dispositivos y facilita a los desarrolladores y proveedores de contenidos adaptar sus servicios con información en tiempo real de la propia red.

Surgen tecnologías que permiten la segmentación de la red como el *network slicing* que permiten construir múltiples redes lógicas de una forma más flexible, dinámica y escalable para adaptarse a necesidades de servicio diferentes sobre una infraestructura física común compartida. Esta capacidad para adaptarse a un amplio rango de requisitos técnicos permite a las redes 5G ofrecer los servicios básicos contemplados en el estándar 3GPP (uRLLC, eMBB y mMTC) y otros definidos específicamente para las necesidades de sus usuarios. Entre estos requerimientos específicos se

pueden encontrar: ancho de banda, QoS, latencia, disponibilidad, seguridad y muchos otros dependientes de cada tipo de servicio. Esto permite también mejorar el factor de utilización de los segmentos (*slices*) de la red, ya que son utilizados de forma simultánea para ofrecer servicio a diferentes usuarios.

Los modelos basados en teoría de colas han ayudado a analizar el rendimiento de sistemas y la evolución de parámetros de QoS considerando cada elemento de forma aislada. Las redes de colas consisten en múltiples colas con uno o más servidores que se pueden relacionar unas con otras en un grafo y donde se producen procesos estocásticos tanto en el proceso de llegada como en los servidores. Esta característica hace a los modelos basados en redes de colas muy interesantes para estudiar en las redes 5G cómo desplegar servicios virtualizados basados en VNFs. El rendimiento del sistema completo puede ser estimado a partir de modelos basados en redes de colas. Se pueden estudiar problemas de ubicación de recursos computacionales (VNF), cumplimiento de restricciones de latencia en la provisión del servicio, probabilidades de bloqueo de servicio, throughput, número de respuestas por servicio, tiempos de respuesta medio, tiempos medios de espera y tasas de utilización de un servidor.

2.3.3. Modelado de servicios de Misión Crítica sobre redes B5G

Las redes 5G pueden ofrecer muchos servicios diferentes a los usuarios aprovechando las capacidades de referencia que proporciona los tres extremos del "triángulo 5G": aplicaciones ultra fiables de baja latencia (uRLLC), comunicaciones masivas de dispositivos (IoT) y banda ancha mejorada (eMBB). Algunos ejemplos son servicios para vehículos autónomos, gestión de tráfico e infraestructura IoT en ciudades inteligentes, automatización industrial, realidad aumentada y realidad virtual.

Dada las ventajas técnicas de las redes 5G y su alta confiabilidad, estas son muy adecuadas para soportar los servicios de misión crítica. Se trata de servicios donde puede haber riesgo de vidas humanas. Por ejemplo: (a) servicios de emergencia como bomberos y rescatistas, (b) servicios de seguridad pública como policía, fuerzas armadas y (c) servicios de telemedicina para diagnóstico remoto o cirugías a distancia.

El grupo NQaS tiene una muy fuerte experiencia en la investigación sobre servicios de misión crítica y en especial en la tecnología de MCPTT. Fruto de esa experiencia se ha creado una spin-off (Nemergent Solutions S.L.) que está desarrollando tecnología MCPTT de vanguardia a nivel internacional. Desde hace muchos años el grupo NQaS participa en los organismos de estandarización de ETSI y 3GPP en los grupos relacionados con estas tecnologías. En especial, colabora con el grupo de trabajo SA6 de 3GPP que es el organismo que inició la estandarización de MCPTT sobre LTE en la versión 13 y ha agregado nuevas funcionalidades en las versiones posteriores para incluir, aparte de los servicios base de voz, los servicios de transmisión de video (MCVIDEO) y datos (MCDATA).

Tradicionalmente, las comunicaciones de ambulancias, bomberos y policía se realizan por redes propias específicas como TETRA, P25, TETRAPOL o DMR. Las comunicaciones entre personas se realizan con tecnología Push-to-Talk, que corresponde con la forma de funcionar tipo walkie-talkie tradicional, es decir, un interlocutor que quiere hablar debe pulsar un botón para pedir el control del canal y los demás escucharán su mensaje por ese canal. El estándar impulsado por el 3GPP denominado MCPTT está pensado para migrar este tipo de servicios para que funcione sobre una red celular 4G-LTE, 5G o B5G. Ampliaciones de este estándar incorporan al servicio posibilidad de intercambiar vídeos (MCVIDEO) y envío de datos (MCDATA). Se trata pues, que las personas involucradas en servicios de misión crítica, como ambulancias, bomberos y policía, pasen de utilizar terminales clásicos a usar dispositivos avanzados B5G en infraestructuras de redes celulares B5G y que su forma de operar, en esencia, siga siendo similar a la que durante años ha demostrado ser eficaz para este tipo de emergencias.

MCPTT soporta dos modos de funcionamiento, con la red o sin la red. En el caso de estudio de este trabajo, se estudiará el modo de funcionamiento con red 5G o B5G. Para este modo, el servicio MCPTT se debe proveer como un servidor MCPTT centralizado accesible via IP tal y como se hace para otro tipo de Servicios de Aplicación de la infraestructura 5G. Para coordinar todo esto hay que considerar la infraestructura de señalización *IMS (IP Multimedia Subsystem)*, pues es la que usan los operadores de las redes 5G para gestionar las llamadas a servicios *AS (Application Services)* y la incorporación de los dispositivos *UE (User Equipment)* involucrados en una comunicación. Hay que indicar que las comunicaciones en MCPTT pueden ser entre grupo de usuarios.

En una entidad MCPTT se gestionan dos componentes, el control de la llamada (*call control*) y el arbitrio de la comunicación (*floor control*). El control de la llamada es el encargado de coordinar y gestionar la asignación de recursos, la gestión de canales la información de usuarios, la afiliación a grupos de usuarios y los tipos de llamada. Se soportan diferentes tipos de llamada tales como: (a) llamadas a grupo básicas, (b) llamadas privadas y (c) llamadas a grupos de difusión (broadcast). Las llamadas a grupo se pueden establecer entre dos o más usuarios adscritos a un grupo particular de MCPTT, mientras que las llamadas privadas se producen entre dos usuarios cualesquiera sin importar su afiliación a grupos. El control de la llamada soporta tres tipos de llamada: básica, peligro inminente y emergencia. Cada tipo supone un estatus diferente y tiene asociados recursos diferentes. Por otro lado, el arbitrio de la comunicación, conocido como *floor control*, es el responsable de establecer quién puede hablar durante una llamada en curso. Hay que entender que esta operación condiciona cómo se producen los flujos de comunicación entre los participantes en una situación determinada.

En la Figura 2.7 se puede ver cómo se despliega la arquitectura de un servidor MCPTT sobre una infraestructura 5G de un operador. Se pueden distinguir los componentes que participan en un *MCPTT Application Server (MCPTT AS)* de los que cabe destacar la función de participación o

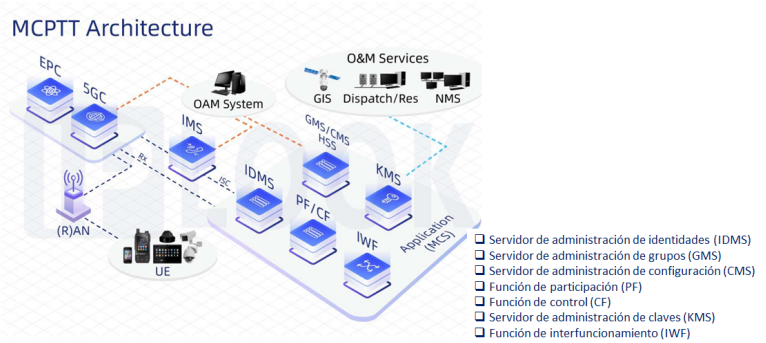


FIGURA 2.7: Arquitectura MCPTT. Fuente: [98].

Participating Function (PF) y la función de control o *Controlling Function (CF)*. La primera de ellas, el participante, es el módulo que interacciona con las entidades externas del operador de red 5G, bien para señalizar con los dispositivos que participan en las comunicaciones MCPTT, proceder a la afiliación a grupos de usuarios y gestionar los mensajes de control de llamadas y petición de canal (*floor control*). La segunda función, el controlador, se encarga de gestionar internamente en el servidor el control de la llamada, interactúa con el GMS para obtener el estado de afiliación a grupos y gestiona el uso del canal de comunicaciones en llamadas a grupos o privadas. Es el que tiene constancia de los participantes en una llamada a grupo de cara a gestionar los flujos de datos a transmitir.

En la Figura 2.8 se puede ver un ejemplo de comunicación MCPTT entre dos terminales y sus interacciones de señalización IMS con los componentes de la red 5G.

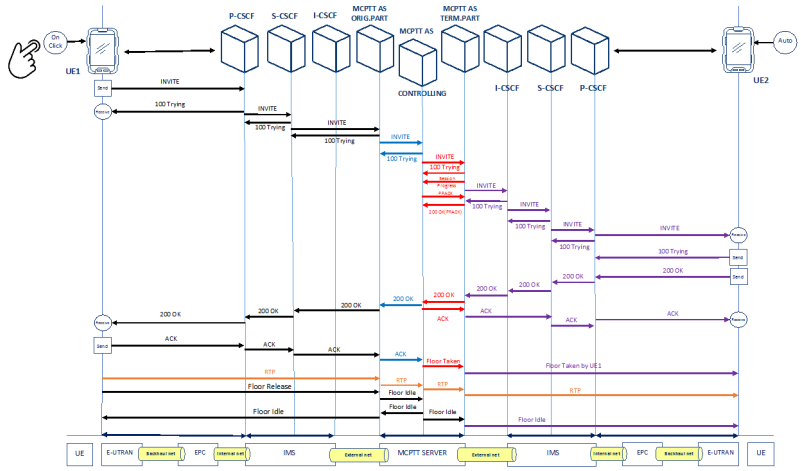


FIGURA 2.8: Ejemplo de comunicación MCPTT. Fuente: [99].

Sirva todo esto para intentar entender cómo se produce una comunicación MCPTT entre dos terminales de usuario en una red 5G. Hay que aclarar que la mayor parte de lo que se ha visto corresponde al plano de control donde, básicamente, se han intercambiado mensajes de señalización de tipo *SIP (Session Initiation Protocol)*. Quedaría, por tanto, aclarar lo que ocurre en el plano de usuario. En la gráfica de la sesión MCPTT, esa etapa correspondería a lo que se indica como tráfico *RTP (Real Time Transport Protocol)* y de *floor control*, que realmente está implementado como tráfico *RTCP (Real Time Transport Control Protocol)*, con los dispositivos. Es de destacar que la distribución del flujo de voz entre los participantes no se realiza directamente de terminal a terminal, sino que necesariamente se realiza a través del servidor MCPTT AS. Esto es así, porque de esa forma lo define el estándar.

Llegados a este punto, estamos en condiciones de plantear cuál es la problemática de distribución del servicio MCPTT o, en general, MCX si consideramos la incorporación al servicio de la transmisión de vídeo (MCVIDEO) y datos (MCDATA). Si se contempla cómo sería la distribución de ese servicio para una comunicación MCPTT en grupo, con participantes situados en diferentes celdas, gestionados por gNodes diferentes, se podría ver que los flujos generados por el participante que tiene el *floor control* estarían dirigidos al servidor MCPTT AS y este los replicaría a cada uno de los participantes en otros puntos de la red 5G. En el plano de control, las interacciones de la señalización IMS pueden estar justificadas, pero en el plano de usuario se produce una concentración de tráfico sobre el servidor MCPTT AS que puede provocar problemas de red con desviaciones significativas de los indicadores de rendimiento clave KPIs. Realmente, ateniéndose al procedimiento actualmente estandarizado, esta problemática no es solucionable. Sin embargo, el grupo NQaS está desarrollando investigación para poder proponer a los organismos de estandarización soluciones de futuro que permitan aliviar esta problemática.

Si consideramos el potencial de las redes 5G, no es difícil entender que tecnologías definidas anteriormente como MEC, permitirían desplazar algunas funciones críticas de la arquitectura MCPTT a los extremos de la red o incluso a centros de servicio intermedio alojados en sitios estratégicos de la infraestructura. La virtualización de servicios de la arquitectura MCPTT en VNFs facilitaría la distribución inteligente de esta arquitectura en diferentes localizaciones próximas a los usuarios que demandan estos servicios en función de la disponibilidad de recursos computacionales próximos a los nodos de la red en los extremos. Algunos estudios realizados dentro del grupo NQaS [100] han demostrado que la ubicación de los usuarios del servicio MCPTT es importante para distribuir el servicio con garantías de QoS.

La teoría de colas puede ayudar en la planificación adecuada de las arquitecturas MCPTT. No obstante, también es aplicable en muchos otros sentidos en la investigación de los rendimientos en la provisión del servicio MCX o en el estudio de algoritmos de planificación en el control de canal, para el caso de encolar las peticiones de los usuarios en una comunicación de grupo.

2.4. Revisión de modelos de colas aplicados a entornos 5G

Siguiendo la clasificación propuesta en la tesis respecto a las líneas de estudio abiertas en la investigación sobre el modelado de servicios en redes 5G usando teoría de colas, se van a presentar los trabajos más destacables que se pueden encontrar en el estado del arte.

Primeramente, se van a revisar algunos trabajos orientados al modelado de sondas para la captura de tráfico en redes 5G. En algunos casos se centran en estudiar las problemáticas de estos sistemas en la interfaz RAN. La parte aérea es una de las etapas con mayor probabilidad de pérdida de paquetes y degradación potencial de la comunicación. Los autores de [101] estudian algoritmos de gestión de colas para el protocolo *Radio Link Control (RLC)* para mejorar el rendimiento en cuanto a latencias, pérdida de paquetes y uso del ancho de banda.

Los autores de [102] estudian cómo evoluciona el retardo del tráfico extremo a extremo o *End-to-End (E2E)* dentro de un gNodeB incluyendo la parte Core y la interfaz aérea RAN. En este trabajo se referencian otras investigaciones [103][104][105] para estimar los retardos en redes 5G que utilizan modelos basados en teoría de colas que permiten analizar los tiempos medios de espera en función del proceso de llegada de paquetes y la forma de servicio. Sin embargo, estos resultados difícilmente se ajustan en su distribución probabilística a lo observado en medidas reales sobre la infraestructura. Por eso Fadhil y Oliveira proponen la utilización de un modelo de mezcla gaussiana o *Gaussian Mixture Model (GMM)* que permite representar de mejor forma la distribución estadística del retardo E2E.

Algunos trabajos se centran en estudiar el rendimiento de las sondas de tráfico en su aplicación en sectores verticales concretos. Jianfang y otros [106] aplican teoría de colas para estudiar el rendimiento de las comunicaciones inalámbricas en redes celulares 5G para el sector vertical industrial de comunicaciones dispositivo contra dispositivo o *Device to Device (D2D)*. Utiliza una distribución estocástica geométrica para situar los dispositivos y a partir de ahí calcular la tasa de señal frente a interferencia y ruido (*Signal to interference-plus-noise rate-SINR*) que se relaciona con la probabilidad de transmisión del dispositivo. Plantean un modelo dinámico de tráfico basado en simular la transmisión con una cola M/G/1 que utilizan para evaluar los rendimientos del sistema. Esto les permite obtener throughputs, retardos medios y probabilidad de pérdida de paquetes, así como valores medios de longitud de las colas. Otros trabajos previos [107] aplican modelos similares en el sector de comunicaciones entre vehículos.

La capacidad de virtualización que se puede realizar en la parte RAN de una red 5G permite que se pueda disponer de componentes desagregados que pueden ser conectados a través de interfaces abiertas. Esto permite disponer de controladores inteligentes que pueden optimizar el uso de los recursos en la interfaz RAN. Esta propuesta está siendo desarrollada por

una alianza de la industria de telecomunicaciones en una especificación conocida como *Open-RAN (O-RAN)* [108]. La arquitectura O-RAN incorpora controladores inteligentes *RIC (RAN Intelligent Controler)*) que pueden tomar decisiones de planificación de recursos en la interfaz RAN. Para ello propone la utilización de técnicas de Inteligencia Artificial (IA) y aprendizaje automático o *Machine Learning (ML)* . En el grupo de investigación NQaS se están desarrollando investigaciones para proponer incorporar modelos basados en teoría de colas para ayudar a estos sistemas en la toma de decisiones adecuada en tiempo real. Dichos modelos de colas pueden plantear escenarios de planificación de recursos que pueden ser evaluados dinámicamente mucho más rápido y más adaptado a las condiciones reales de lo que se puede conseguir con técnicas de IA.

En segundo lugar, se pueden encontrar trabajos que utilizan modelos basados en redes de colas para el modelado de sistemas de provisión de servicio E2E en redes 5G.

Algunas propuestas incorporan la capacidad de dotar de recursos computacionales a los nodos de la red 5G con tecnologías como MEC y CE-RAN. Esto se justifica debido al crecimiento explosivo del volumen de tráfico en la red, el enorme número de dispositivos conectados y el uso de aplicaciones multimedia. Las redes 5G deben proporcionar estas capacidades e incluso para aplicaciones que demandan baja latencia. Surge así el concepto de *Network Slicing* que utiliza NFV y tecnologías como SDN para obtener prestaciones adaptadas a las necesidades de los usuarios de la red en la gestión del tráfico.

Con la segmentación de red, hasta se puede pensar en ejecutar instancias de la interfaz RAN como un trozo (*slice*) más virtualizado. Es lo que se denomina *Cloud RAN (C-RAN).* AlQahtani y Alhomiqani [109] proponen en su trabajo la utilización de un modelo basado en teoría de colas para estimar extremo a extremo los parámetros de QoS en una infraestructura 5G con una serie de servicios virtualizados para RAN y MEC. Se consideran diferentes segmentos (*slices*) para diferenciar los servicios básicos de 5G como uRLLC, eMMB y mIoT. El modelo puede ser usado para estudiar la escalabilidad de la red de forma dinámica, el consumo de CPU y los parámetros típicos de rendimiento de los modelos de colas.

El caso más común de los trabajos de investigación de este apartado es el de utilizar modelos de colas para simular el comportamiento de una entidad virtual asociada a una VNF. En general se conoce la forma de realizar esas funciones virtuales extremo a extremo y es posible definir una cadena de VNF involucradas en el desarrollo de un servicio. Algunas tecnologías como SDN permiten identificar estas relaciones donde la cadena de VNF es gestionada por un controlador de virtualización que las coloca en un substrato de red física dentro de un nodo NFV con sus enlaces virtuales.

Se ha creado una base teórica interesante en la resolución de modelos que utilizan cadenas de provisión basadas en VNFs y que tienen su aplicación para resolver problemáticas en redes 5G. Se pueden destacar entre estas problemáticas: (a) estudio de parámetros de calidad QoS extremo a

extremo (b) dimensionamiento de recursos (c) despliegue y ubicación óptima de recursos y (d) rutado de tráfico. Aparece el concepto de servicios de red *softwarizados* (*softwarized network services*). La teoría de colas está demostrando su utilidad para resolver estos problemas basados en modelos de redes de colas.

Un trabajo que ejemplariza muy bien el fundamento de esa base teórica es el de los autores de [110]. Presenta una arquitectura de una entidad de gestión de movilidad virtualizada o o *virtual Mobility Management Entity (vMME)*. Plantea un modelo de colas de red abierta con nodos G/G/m que permite representar la cadena de provisión de VNF y utiliza técnicas clásicas de evaluación, como es el método de análisis de valor medio, o *Mean Value Analysis (MVA)*, y lo compara con los resultados obtenidos por un analizador de redes de colas, *Queueing Network Analyzer (QNA)* según la técnica propuesta por Whitt [111].

En [105] se propone estudiar el retraso extremo a extremo en una cadena de provisión de VNFs que utiliza un modelo de cola basado en la estrategia *Dominant Resource Generalized Processing Sharing (DR-GPS)* que permite estudiar cómo diferentes flujos de tráfico comparten recursos en cada nodo NFV. Los recursos compartidos son la CPU y la capacidad de los enlaces. Este trabajo permite, gracias a esa estrategia de cola DR-GPS, considerar en la cadena de provisión de VNF los retardos no solo provocados por los enlaces físicos, sino también por las necesidades computacionales de cada flujo. En teoría de colas existen muchas estrategias diferentes de atención a una cola y cada una de ellas puede ser adecuada a diferentes tipos de tráfico.

Sobre la problemática de ubicación óptima de recursos en redes 5G, es de destacar el trabajo de Agarwal *et al.* [112] que plantea cómo en estas redes, cuando deben ajustarse a los requerimientos de diferentes sectores verticales (por ejemplo, Industria) es necesario decidir de la forma más automática posible cuestiones como la ubicación de funciones VNF, la asignación de recursos y el rutado de tráfico. Para ello propone un modelo basado en redes de colas.

Dentro del grupo de investigación NQaS, se han realizado trabajos sobre ubicación óptima de recursos. En [113], se plantea el problema como una formulación de optimización matemática con un conjunto de parámetros y condiciones de contorno que representan las restricciones impuestas por la infraestructura física y las limitaciones del servicio. Este problema no se resuelve con teoría de colas, sino que se recurre a técnicas propuestas por la teoría de optimización de redes neuronales. En concreto, se usa la técnica de Aprendizaje Reforzado o *Reinforcement Learning*. El sistema es capaz de aprender cómo tomar decisiones de ubicación de recursos para minimizar el consumo de energía. Este trabajo ha servido para contextualizar la investigación del grupo y como continuación de él se han propuesto modelos basado en teoría de colas que permiten introducir en el problema modelos estocásticos con distribuciones de servicio más ajustadas a sistemas reales y con la posibilidad de introducir parámetros de medida de la QoS.

Por último, en lo referente al modelado de servicios de Misión Crítica

MCPTT, las referencias son pocas y de no muy alto valor en cuanto al uso de teoría de colas.

En el año 2019, se presentó una tesis doctoral [114] en el Instituto Tecnológico de Florida que utiliza modelos básicos de teoría de colas. Es el único trabajo que se ha encontrado que usa teoría de colas para el modelado del servicio MCPTT sobre redes celulares. Utiliza modelos para la estimación de rendimiento en el bloqueo de llamadas de grupo, duración de los tiempos de bloqueo, estudio de los *timeout* para llamadas de grupo y pocas cosas más de interés. Este trabajo, aunque hace referencia al uso de la tecnología sobre redes celulares, en los modelos, no se contempla nada sobre el tema y se centra más sobre aspectos básicos de la operativa de usuarios en este tipo de servicios.

Es de destacar el trabajo de los autores de [115] en el desarrollo de un módulo de extensión del simulador ns-3 que permite evaluar el rendimiento del servicio MCPTT sobre un modelo creado en base al estudio de una muestra significativa de llamadas reales. Se presenta el diseño de la herramienta y proporciona una API abierta que pueda ser extendida. Resulta muy interesante para crear escenarios de prueba y poder validar resultados.

En este sentido, ha sido de interés la tesis de Sanchoyerto [99], desarrollada dentro del grupo NQaS, donde se proporciona información sobre tiempos de servicio medio en las interacciones del protocolo MCPTT obtenidas de analizar trazas de sistemas desplegados en tiempo real y que pueden servir para parametrizar modelos propuestos en esta tesis. En el trabajo de Sanchoyerto, se pueden ver estudios de los retardos acumulados en las diferentes partes de la red 5G en la prestación del servicio MCPTT. Se analizan también problemáticas de comunicaciones en grupo como las que se propone modelar en esta tesis.

Para acabar con los servicios de misión crítica, resulta muy interesante estudiar el trabajo [116] sobre la arquitectura distribuida del servicio MCPTT. Esta distribución del servicio se basa en la capacidad MEC de los gNodeBs y permite colocar el plano de usuario en los extremos de la red, mientras se mantiene el plano de control centralizado sobre el MCPTT AS. En esencia, es la base tecnológica de lo que se propone modelar en esta tesis. Hay que recordar que la estandarización de 3GPP actualmente no permite que se separe el plano de usuario del servidor centralizado MCPTT AS, pero eso es en lo que se está trabajando ahora desde el grupo NQaS, para proponer modificaciones que lo permitan.

Para terminar esta revisión de trabajos relacionados con modelos de colas sobre redes 5G, se citan brevemente algunos más. En [117], se presenta un modelo de colas para controlar el número de *User Plane Functions* (UPF) en un entorno de red 5G; el sistema es descrito por una cadena de Markov de tiempo continuo bidimensional. Los autores de [118] desarrollan un modelo analítico de colas pra analizar las estrategias de *network slicing* en redes 5G; la solución y las métricas de rendimiento del sistema se derivan a partir de una cadena de Markov bidimensional. Los modelos analíticos de [119] y [120] estiman los recursos computacionales requeridos para cumplir los objetivos de calidad de servicio de algunas aplicaciones

de monitorización en la nube; son modelos basados en colas M/M/1 y M/M/c/K. En [121], el campo de aplicación son las redes vehiculares ad hoc o *Vehicular Ad Hoc Network* (VANET) y se estudia cómo calcular varias métricas de rendimiento (tiempo de espera del sistema, throughput o número de recursos requeridos para satisfacer las necesidades de los usaurios) mediante dos modelos de teoría de colas denominados M/M/c y M(N)/M/c. Por su parte, [122] explica cómo modelar los bloques que constituyen un nodo 5G. También hay propuestas de modelado relacionada con arquitecturas SDN. En [75], se evalúa la calidad de la operación SDN estimando los valores medios del retardo de paquetes con colas M/G/1 y G/G/1. Y en [76], se propone un modelo de Markov basado en una cola M/M/c para gestionar el tráfico rutado entre los nodos de acceso a la red 5G y la arquitectura SDN.

2.5. Conclusiones

Este capítulo sobre el estado del arte se ha dividido en dos bloques. En el primero, se han examinado los sistemas de captura y análisis de tráfico de red basados en hardware de propósito general y se ha podido ver que existen numerosas propuestas de este tipo de arquitecturas. Se pueden distinguir tres etapas en orden cronológico:

- Una primera etapa con propuestas de sondas sobre redes de 1 y 10 Gbps donde existen verdaderas dificultades para capturar y analizar los paquetes a velocidad de línea.
- Una segunda etapa, también sobre redes de 1-10 Gbps, en la que las sondas obtienen mejores resultados de rendimiento. Esto se debe principalmente a la mejora que experimentan los equipos con hardware de propósito general en cuanto a memoria, procesamiento multicore, tarjetas de red, etc.
- Una tercera etapa, donde las redes son de mayor velocidad, aparece la red 5G y los entornos virtualizados. En este contexto, los agentes de monitorización propuestos se basan en funciones VNF.

En la segunda parte del bloque sobre sondas de análisis de tráfico, se hace una revisión de los estudios analiticos y de modelado realizados sobre este tipo de sistemas. Entre los que se identifican, destacan los basados en teoría de colas; algunos de ellos pueden tener interés para que sean adaptados y aplicados a la investigación relacionada con este trabajo. No obstante, hay que decir que los trabajos de modelado sí existen, pero que no son numerosos, si se compara con el número de propuestas de arquitecturas. Esto confirma el interés de este trabajo en realizar aportaciones en materia de modelado aplicado a sondas de análisis.

A pesar de que esta tesis se centra en aspectos de modelado, no hay que descuidar la parte experimental. Por ello, en el caso de este trabajo, es importante tener en cuenta la experiencia del grupo de investigación NQaS que ya dispone de un prototipo de sonda de análisis de tráfico llamada

Ksensor [40]. La sonda puede servir para validar modelos; también puede ayudar en la construcción de nuevos modelos. En este punto, analizada la situación, y conociendo que han surgido problemáticas relacionadas con la sonda Ksensor en pruebas de laboratorio experimentales, se decide realizar un trabajo de modelado de la sonda orientado a los siguientes puntos:

- Mejorar el rendimiento de la sonda, ya que en pruebas de laboratorio se ha observado un throughput del sistema menor del esperado.
- Analizar el sistema de captura de Linux y evaluar su rendimiento mediante modelado. Para esta tarea, se tendrá en cuenta el trabajo preliminar [123] del autor de esta tesis.

Por tanto, la conclusión que se extrae del primer bloque de este capítulo es que en este trabajo de tesis se abordan esas dos tareas de modelado que serán desarrolladas en los capítulos 3 y 4 del documento.

El segundo bloque del capítulo presenta el cambio que se quiere dar hacia una nueva línea de investigación llamada "Despliegue y orquestación inteligente de servicios en entornos 5G, B5G y 6G". Se pretende aplicar el conocimiento adquirido en el modelado de sondas con teoría de colas sobre la nueva línea. Por ello, se identifican dos posibles campos de aplicación: por un lado, las sondas de captura y análisis de tráfico sobre red 5G (que utilizan tecnologías como NFV y SDN); por otro lado, el modelado de la provisión de servicios extremo a extremo en redes 5G, considerando que para ello es necesario realizar el despliegue de funciones VNF sobre la red. Dada la experiencia del grupo NQaS en los servicios de Misión Crítica, se selecciona la segunda opción y se decide modelar un servicio de este tipo sobre red B5G.

En la segunda parte de este bloque sobre servicios de redes B5G, al igual que se ha hecho en el primer bloque, se hace una revisión de trabajos previos sobre modelado basado en teoría de colas aplicado a entornos 5G. Aquí también se identifican varios trabajos y se confirma que la teoría de colas es aplicable en este campo.

Con todo ello, al finalizar este segundo bloque, se concluye que este trabajo de tesis va a hacer una tercera propuesta de modelado basada en teoría de colas dirigido a la siguiente cuestión:

- Desplegar de manera eficiente un conjunto de funciones VNF sobre una red B5G, para proveer un servicio de Misión Crítica que incluye tráfico de voz Push-To-Talk, datos y vídeo.

Esta tercera propuesta de modelado será desarrollada en el capítulo 5 del documento de tesis.

Capítulo 3

Modelo de optimización de throughput basado en proceso de decisión de Markov

El capítulo sobre el estado del arte finaliza identificando los sistemas susceptibles de ser modelados en esta tesis. Entre ellos están las sondas de análisis de tráfico de red y, en este capítulo, se desarrolla la primera propuesta de modelo matemático de esta tesis sobre ese tipo de sistema. El trabajo relacionado con este primer modelo surge porque, en un escenario de investigación donde se tiene una sonda de análisis de tráfico de red basada en Linux e instalada sobre equipamiento hardware de propósito general, se identifica un problema de pérdida de rendimiento de dicho sistema de captura y análisis de tráfico de red. Esto se detecta a través de unas medidas experimentales de throughput tomadas sobre la propia sonda. Dada esta situación inicial, el objetivo de este capítulo es desarrollar un modelo que optimice el rendimiento de dicho dispositivo que captura y analiza tráfico de red. La mejora que se pretende lograr será medida en términos de throughput.

El capítulo sigue la estructura que se detalla a continuación. El apartado 3.1 introduce el entorno de investigación para el que se desarrolla el modelo e identifica las problemáticas observadas. Seguidamente, el apartado 3.2 describe el modelo que se trata de una cola tándem que se convierte en un modelo de tiempo discreto mediante técnicas de uniformización. Posteriormente, en el apartado 3.3 se formula el problema de optimización mediante un proceso Markov Decision Process (MDP) y se resuelve. A continuación, el apartado 3.4 evalúa el modelo para un conjunto de valores de entrada relacionados con el entorno de pruebas que posteriormente se utilizará para validar el modelo. Mediante simulación se obtiene el rendimiento de la política óptima resultante del proceso MDP planteado. Finalmente, el apartado 3.5 presenta la validación del modelo. Para ello, sobre la sonda

real, se implementa la política óptima (o una próxima a la óptima) y se comprueba si se produce la esperada mejora de throughput.

3.1. Introducción. Contexto inicial

Como ya se ha explicado en el capítulo sobre el estado del arte, con los avances del hardware de propósito general o *commodity hardware*, surgieron varias arquitecturas software de captura y análisis de tráfico de red [124]. También se ha indicado que muchas de esas propuestas funcionaban sobre Linux por tratarse de un sistema operativo abierto, modular y flexible. Dentro de esas iniciativas, también estaba la sonda llamada Ksensor [40] del grupo de investigación NQaS que, sobre una máquina de propósito general, realiza la captura y el análisis en el núcleo (o kernel) de Linux. Es en este entorno donde, a través de unas medidas experimentales sobre el prototipo de la sonda, se observa un comportamiento no deseado, ya que el rendimiento del dispositivo es menor que el esperado. Antes de describir dichas observaciones, en el siguiente punto, se recuerda brevemente el esquema de recepción de paquetes de Linux. Tras esto, se detallará la problemática que pretende resolver el modelo que es objeto de estudio en este capítulo.

3.1.1. Proceso de recepción de paquetes en Linux

En el caso de una sonda basada en Linux, la trayectoria que sigue un paquete desde que entra en el sistema hasta que es entregado a la aplicación de monitorización puede representarse tal y como se muestra en la Figura 3.1 [87]. Se distinguen tres etapas:

- En primer lugar, el paquete pasa de la tarjeta NIC al búfer circular (o *ring buffer*). Dado que es una transferencia DMA, este movimiento no supone ningún consumo de CPU.
- La segunda fase corresponde a la etapa de captura. Los paquetes almacenados en el búfer circular son tratados a nivel de kernel por el proceso denominado *softirq* [125]. Esto sí conlleva consumo de CPU. Cuando termina este procesamiento, el paquete es extraído del búfer circular y es transferido al búfer de análisis.
- Finalmente, en la tercera etapa, los paquetes almacenados en el búfer de análisis son tratados por la aplicación de monitorización. Para que el esquema resulte más simple, se supone que la monitorización incluye los módulos de filtrado y análisis.

Por tanto, desde el punto de vista de consumo computacional, solamente influyen dos de las tres etapas: la de captura y la de análisis. Además, si se da una situación de recursos limitados y solamente se dispone de un procesador, ambos procesos (captura y análisis) no pueden estar en ejecución al mismo tiempo. Ante esto, resultará de interés ajustar adecuadamente la utilización del procesador para que el rendimiento de la sonda sea óptimo.

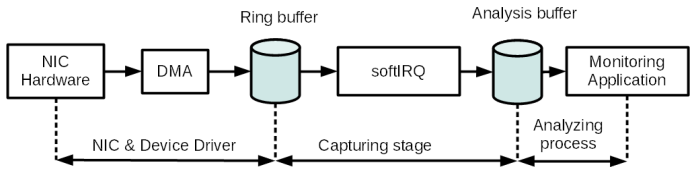


FIGURA 3.1: Proceso de recepción de paquetes en una sonda basada en Linux.

3.1.2. Identificación de las problemáticas en el entorno de investigación

A continuación, se va a explicar la situación de partida que motiva la propuesta del modelo matemático que se verá en los siguientes apartados de este capítulo. La Figura 3.2 muestra la medida experimental de throughput tomada sobre la sonda desarrollada por NQaS. El eje de abscisas representa la tasa de paquetes entrantes a la sonda, medida en paquetes por segundo (pps). En el eje de ordenadas se ve el throughput de análisis que provee la sonda en su salida, esto es, el número medio de paquetes por segundo que es capaz de cursar el sistema en su totalidad. El experimento se hace sometiendo a la sonda a distintas cargas de análisis, referenciadas como *null*, *0.3k* y *1k*. Cada carga de análisis supone tener mayor o menor procesamiento en la etapa de la aplicación de monitorización, siendo el caso *null* el que corresponde a una carga baja (es decir, un tiempo de procesamiento bajo en la etapa de análisis), *0.3k* a una carga media, y *1k* a una carga alta.

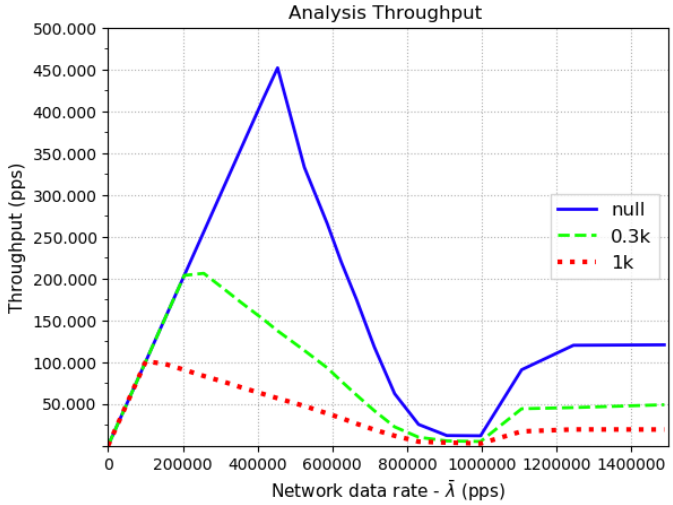


FIGURA 3.2: Rendimiento inicial de la sonda de análisis real.

Se observa que, para todas las cargas de análisis (baja/*null*, media/*0.3k* y alta/*1k*), el comportamiento es similar. Para el rango de tasas entrantes

bajas (de 0 a 450.000 pps aproximadamente) el comportamiento es bueno, ya que el throughput es prácticamente igual a la tasa de tráfico entrante. Sin embargo, a partir de, aproximadamente, 450.000 pps, el throughput cae ostensiblemente y llega a sus valores mínimos con tasas entre 800.000 y 1.000.000 pps; esta pérdida de rendimiento resulta anómala. Por último, en el último rango de tasas (sobre 1.000.000-1.500.000 pps), el throughput mejora y, al final, se mantiene a un valor relativamente constante.

Las causas de este problema del sistema de análisis de tráfico pueden ser las siguientes:

Balance ineficiente de la carga computacional. La arquitectura puede presentar un balance computacional ineficiente entre los procesos de captura y análisis. Por una parte, la captura se realiza en contexto de kernel, mientras que los procesos de análisis generalmente se asocian con aplicaciones de nivel de usuario. La planificación de tareas, sin la apropiada asignación de prioridades, prima a los procesos de más bajo nivel (kernel) frente a las aplicaciones de usuario, al margen de los tiempos de consumo de cada una de las etapas. El incremento de la capacidad de procesamiento del sistema puede ser una solución, aunque en este trabajo de tesis no se contemplará. Para el estudio matemático, resulta de mayor interés el caso de recursos computacionales limitados. En todo caso, resulta igualmente necesario introducir estrategias que aseguren una ejecución eficiente de los procesos de captura y análisis.

Ausencia de coordinación entre las etapas de procesamiento. No existe comunicación alguna entre las etapas de procesamiento (captura y análisis de paquetes). La ejecución de los correspondientes procesos se realiza de manera independiente y no se controlan los recursos consumidos por cada fase. Esta ausencia de coordinación lleva a que, en situaciones de saturación, el sistema pueda estar consumiendo recursos en la captura de paquetes que posteriormente no van a poder ser procesados, debido a que la etapa de análisis ya no dispone de recursos suficientes para ello. Unido a la carencia anterior, la falta de control sobre la captura hace que esta malgaste recursos en tareas que no se traducen en un mayor número de paquetes procesados, sino todo lo contrario: las tareas de captura restan capacidad a las tareas de análisis, que no disponen de recursos suficientes para procesar los paquetes ya capturados; esto hace que cualquier nuevo paquete que entre en el sistema no vaya a poder ser analizado y, por tanto, la capacidad consumida en su captura se habrá desperdiciado. Por ello, es importante introducir elementos de diseño que optimicen la ejecución de las distintas etapas de procesamiento y eviten el consumo innecesario de recursos computacionales, por ejemplo, en la captura de paquetes que no van a poder ser procesados.

Ineficiencia en el procesamiento de paquetes. Las transferencias de información entre espacios de memoria (DMA, kernel y usuario) introducen operaciones de recopia de información que sobrecargan el procesamiento de los paquetes. Estas operaciones conllevan además reservas de memoria,

gestión de búferes, cambios de contexto y llamadas a sistema, cuyo coste computacional es elevado.

Como consecuencia de todo esto, visto ese comportamiento de la sonda con una pérdida de rendimiento tan significativa en el rango intermedio de tasas de datos entrantes, se decide desarrollar un modelo matemático. Este deberá representar el comportamiento de la sonda de análisis y proporcionar resultados que ayuden a mejorar su rendimiento en términos de throughput.

3.2. Modelo de colas

Este apartado describe el modelo de colas que representará la ejecución de los procesos de captura y análisis de paquetes de un sistema de monitorización de tráfico. Como ya se ha mencionado anteriormente, desde el punto de vista matemático, el caso de un sistema con recursos limitados es el más interesante. Por ello, se supone que el sistema solamente tiene un procesador para realizar ambas tareas.

3.2.1. Modelo de red abierta de colas "single-server tandem queue"

El modelo propuesto para representar el proceso de recepción de paquetes de un sistema de monitorización de tráfico (véase Figura 3.1) es una red abierta de colas como la de la Figura 3.3. Se trata de una cola tándem donde las tasas de servicio están relacionadas con los consumos computacionales de las tareas de captura y análisis. Los clientes de las colas son los paquetes provenientes de la red que entran a la sonda.

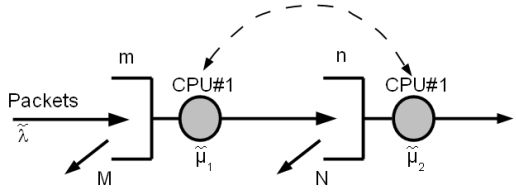


Figura 3.3: Representación mediante un sistema de colas para el caso de un único procesador.

Como se puede ver en la Figura 3.3, los paquetes llegan a la primera cola del modelo. Esta llegada de paquetes corresponde a la transferencia DMA desde la tarjeta de red al búfer en anillo sin que esto suponga ningún consumo de central processing unit (CPU). Por tanto, la primera cola representa el búfer circular de la etapa de captura donde se almacenan los paquetes antes de ser procesados por la *softirq*. El modelo supone que la primera cola, a la que se denominará cola de captura, tiene una capacidad limitada de M paquetes. Si un paquete llega cuando la cola de captura

está llena, dicho paquete es descartado sin ser procesado por el sistema. El tiempo de servicio del primer servidor está relacionado con el tiempo de procesamiento requerido por la CPU para la *softirq*.

Una vez que un paquete es servido por el primer servidor, es decir, se ha completado el procesamiento del paquete en la *softirq*, el paquete pasa a la segunda etapa de la red de colas. Esta está relacionada con la etapa de análisis donde los paquetes esperan en el búfer de análisis hasta que son procesados por la aplicación de monitorización del sistema de monitorización de tráfico de red (véase la Figura 3.1). En el modelo, la segunda cola de espera y el segundo servidor representan el búfer de análisis y el procesamiento de análisis respectivamente. Al igual que la cola de captura, la segunda cola, que también será referenciada como cola de análisis en adelante, también es una cola finita. Tiene una capacidad de N paquetes y, cuando está llena, no pueden entrar los nuevos paquetes que lleguen. Finalmente, tan pronto como el segundo servidor complete el procesamiento de análisis del paquete, este abandonará la red de colas. Esto representa que la aplicación de monitorización ha completado la tarea relacionada con este paquete.

Con objeto de que la solución analítica no se vuelva inmanejable, se supone que la llegada de paquetes a la cola de captura sigue un proceso de Poisson con tasa $\tilde{\lambda}$. También se supone que tanto el tiempo requerido por el procesador para completar el proceso de captura relacionado con cada paquete y el tiempo de análisis de cada paquete en la segunda etapa siguen variables aleatorias exponenciales de media $1/\tilde{\mu}_1$ y $1/\tilde{\mu}_2$ respectivamente. Se conoce que, para ciertos tipos de tráfico como el de redes Ethernet, el proceso de llegadas no sigue un proceso de Poisson, sino que es más un tráfico a ráfagas [126]. No obstante, el caso de estudio es ligeramente distinto, ya que los paquetes entrantes no llegan directamente de la red Ethernet, sino que lo hacen a través de la transferencia DMA. Con respecto a los tiempos de servicio, aunque la ejecución de una función puede tener un comportamiento bastante determinístico, se supone que se introduce cierta aleatoriedad por el tráfico entrante, por la longitud variable de los paquetes y por la incertidumbre asociada al planificador del núcleo del sistema operativo. Por tanto, a pesar de tener estas limitaciones, se mantienen estas suposiciones que simplifican la resolución del modelo. Al final del proceso de modelado, habrá que comprobar la validez de la propuesta.

La red de colas consta de dos servidores, sin embargo, como el caso que se considera es el del sistema con único procesador, esta CPU debe dividir sus esfuerzos entre las dos tareas (captura y análisis). Por esta razón, en el modelo, los dos servidores de la red de colas (ambos denominados *#CPU1* en la Figura 3.3) no pueden estar activos al mismo tiempo. Por tanto, se puede considerar que el sistema consta de una cola tándem con 2 etapas y un servidor móvil. El procesador será asignado a la cola de captura o a la cola de análisis en función de las preferencias del planificador del sistema. Este es el responsable de tomar la decisión.

El modo en el que el planificador del sistema elige la acción correcta para el procesador dinámico tiene un gran impacto en el rendimiento de

la red, que será medido en términos de throughput de análisis. Una asignación demasiado larga para la cola de análisis podría traer la pérdida de paquetes en la cola de captura, mientras que una asignación demasiado larga para la cola de captura podría disminuir la capacidad de análisis de paquetes. Por ello, se pretende determinar cómo debe proceder el sistema en la planificación entre la etapa de captura y la de análisis con el objetivo de optimizar el rendimiento. Para ello, se plantea un análisis basado en un proceso MDP tridimensional que permita obtener una política dinámica que determina cómo se debe asignar el procesador en cada momento. Esta decisión dependerá del número de paquetes en cada cola y de la *posición* del procesador cuando se toma la decisión.

3.2.2. Modelo equivalente en tiempo discreto

El siguiente paso es construir un modelo en tiempo discreto que describa la dinámica del modelo en tiempo continuo presentado anteriormente. Esto se realiza mediante la técnica de uniformización [127]. El modelo en tiempo discreto provee un entorno que facilita la optimización posterior del throughput. Aplicando argumentos de uniformización, se definen los siguientes valores probabilísticos:

$$\lambda = \frac{\tilde{\lambda}}{\tilde{\lambda} + \tilde{\mu}_1 + \tilde{\mu}_2}. \tag{3.1}$$

$$\mu_1 = \frac{\tilde{\mu}_1}{\tilde{\lambda} + \tilde{\mu}_1 + \tilde{\mu}_2}. \tag{3.2}$$

$$\mu_2 = \frac{\tilde{\mu}_2}{\tilde{\lambda} + \tilde{\mu}_1 + \tilde{\mu}_2}. \tag{3.3}$$

Por tanto, se pasa de un sistema continuo a uno discreto donde:

- En cualquier intervalo de tiempo, y siempre que la cola de captura no esté llena (cosa que ocurre con M paquetes en el búfer de la primera etapa), un nuevo paquete llega a la red de colas con probabilidad λ.
- Si el procesador está activo en la cola de captura, y la cola no está vacía, el procesador captura un paquete dentro del intervalo de tiempo con probabilidad μ_1.
- Si el procesador está activo en la cola de análisis, y la cola no está vacía, el procesador termina la etapa de análisis de un paquete dentro de un intervalo de tiempo con probabilidad μ_2.

Las probabilidades que aparecen en las Ecuaciones (3.1), (3.2) y (3.3) son independientes entre intervalos de tiempo diferentes. La longitud de cada intervalo de tiempo, en unidades de tiempo, viene dada por:

$$t_s = \frac{1}{\tilde{\lambda} + \tilde{\mu}_1 + \tilde{\mu}_2}. \tag{3.4}$$

3.3. Formulación del proceso MDP

En este apartado se presenta un problema de optimización orientado a maximizar el throughput de análisis a través del modelo en tiempo discreto descrito en el apartado 3.2.2. Este problema de optimización cae en el entorno de los MDP. Un MDP consta de un espacio de estados, un espacio de acciones y una estructura de recompensa. Siempre que el planificador elige entre las acciones disponibles, se gana una recompensa que depende del estado actual del sistema, y después de recibirla, el sistema evoluciona del estado actual a uno nuevo. El objetivo del planificador es encontrar la secuencia de acciones correcta, maximizando la recompensa total esperada que el sistema llega a obtener.

Se plantea un modelo de MDP en tiempo discreto y tridimensional. Se considera un sistema dividido en intervalos de tiempo $t \in \mathcal{T} := \{0, 1, 2, \dots\}$, en los cuales el sistema puede tomar decisiones. El instante de tiempo t corresponde al comienzo del periodo de tiempo t. En todo t, se toma una decisión para elegir entre dos acciones: asignar el procesador a la cola de captura o a la cola de análisis. El estado del sistema se representa con el número de paquetes en ambas colas y la *posición* del procesador justo antes de tomar la decisión.

A continuación, se proporcionan los elementos que describen la evolución del sistema:

- $\mathcal{A} := \{C, A\}$ es el conjunto de acciones del sistema. Seleccionar la acción C representa asignar el procesador a la primera cola para capturar paquetes, mientras que optar por la acción A conlleva asignar el procesador a la segunda cola para analizar paquetes.

- $\mathcal{S} := \mathcal{M} \times \mathcal{N} \times \mathcal{A}$ es el espacio de estados del sistema. Si el estado del sistema es $s = (m, n, a)$, siendo $m \in \mathcal{M} := \{0, 1, \dots, M\}$, $n \in \mathcal{N} := \{0, 1, \dots, N\}$ y $p \in \mathcal{A}$, esto quiere decir que hay m paquetes en la cola de captura, n paquetes en la cola de análisis, y que la acción previa tomada por parte del sistema ha sido p, por lo que la *posición* actual del procesador es el valor de p.

- R_s^a es la recompensa esperada en un periodo o intervalo si se elige la acción $a \in \mathcal{A}$ cuando el sistema se encuentra en el estado $s \in \mathcal{S}$.

- $P^a := \left(p_{s,s'}^a\right)$ es la matriz de transiciones de estado si se toma la acción $a \in \mathcal{A}$ al comienzo del intervalo. Si $s = (m, n, p)$ y $s' = (m', n', p')$, la probabilidad de transición del estado s al estado s' es $p_{s,s'}^a$.

3.3.1. Estructura de recompensa

Puesto que el objetivo del planificador es maximizar el throughput de análisis, se establece que se gana 1 unidad de recompensa siempre que un paquete sale de la cola de análisis, esto es, cuando se completa la tarea de análisis de dicho paquete. Además, también se considera que la toma de decisión incurre en un coste siempre que se opta por cambiar la *posición* del procesador. Este coste representa el tiempo que pasa cuando el procesador debe cambiar su *posición*, afectando de manera negativa en el rendimiento del sistema. Este coste por cambio o *switching cost* será denotado como SC. La recompensa esperada en un intervalo R_s^a dependerá del estado actual $s = (m, n, p)$ y de la acción a elegida por el planificador y puede expresarse de la siguiente manera:

$$R^a_{s=(m,n,p)} = \begin{cases} 0, & p = C, a = C, 0 \leq n \leq N, \\ -SC, & p = C, a = A, n = 0, \\ \mu_2 - SC, & p = C, a = A, 1 \leq n \leq N, \\ -SC, & p = A, a = C, 0 \leq n \leq N, \\ 0, & p = A, a = A, n = 0, \\ \mu_2, & p = A, a = A, 1 \leq n \leq N, \end{cases} \qquad 0 \leq m \leq M \tag{3.5}$$

La recompensa es cero cuando (i) el procesador permanece en la etapa de captura o (ii) el procesador permanece en la cola de análisis y esta está vacía. La recompensa es menos el coste de cambio ($-SC$) cuando (i) el procesador se mueve de la cola de captura a la cola de análisis y la cola de análisis está vacía o (ii) el procesador se mueve de la cola de análisis a la cola de captura. La recompensa es igual a μ_2 cuando el procesador permanece en la cola de análisis y esta no está vacía. Finalmente, la recompensa es μ_2 menos el coste de cambio ($\mu_2 - SC$) cuando el procesador se mueve de la cola de captura a la cola de análisis y esta no está vacía.

Obsérvese que el problema de minimizar costes no es equivalente al de maximizar throughput, pero, como se verá más adelante, los resultados obtenidos satisfacen el objetivo perseguido.

3.3.2. Probabilidades de transición

Como se ha mencionado anteriormente, $p^a_{s,s'}$ es la probabilidad de pasar del estado $s = (m, n, p)$ al estado $s' = (m', n', p')$ cuando se toma la acción a al comienzo del intervalo. Por ello, el valor de p, la *posición* del procesador en el estado $s = (m, n, p)$, coincide con la acción a, es decir, $p = a$ en el estado s. Por otro lado, la *posición* p' del estado s' será el valor de la acción que se toma en el comienzo del siguiente intervalo. El valor de p' no afecta a la probabilidad de transición.

Se van a plantear las probabilidades de transición $p^a_{s,s'}$ de forma separada, para cada uno de los casos de a. En primer lugar, para la acción

$a = C$, el estado $s = (m, n, p)$ con $p = a = C$ y el estado $s' = (m', n', p')$,

$$p_{s,s'}^{a=C} = \begin{cases} 1-\lambda & m = 0,\ 0 \leq n \leq N,\ m' = 0,\ n' = n \\ \lambda & 0 \leq m \leq M-1,\ 0 \leq n \leq N,\ m' = m+1,\ n' = n \\ 1-\lambda-\mu_1 & 1 \leq m \leq M-1,\ 0 \leq n \leq N,\ m' = m,\ n' = n \\ \mu_1 & 1 \leq m \leq M,\ 0 \leq n \leq N-1,\ m' = m-1,\ n' = n+1 \\ \mu_1 & 1 \leq m \leq M,\ n = N,\ m' = m-1,\ n' = N \\ 1-\mu_1 & m = M,\ n = 0,\ m' = M,\ n' = 0 \\ 1-\lambda-\mu_1 & m = M,\ 1 \leq n \leq N,\ m' = M,\ n' = n \\ \lambda & m = M,\ 1 \leq n \leq N,\ m' = M,\ n' = n \end{cases} \tag{3.6}$$

En la Ecuación (3.6), las probabilidades asumen que

$$\lambda(1-\mu_1) \approx \lambda \tag{3.7}$$
$$(1-\lambda)\mu_1 \approx \mu_1 \tag{3.8}$$
$$(1-\lambda)(1-\mu_1) \approx 1-\lambda-\mu_1 \tag{3.9}$$

Las expresiones anteriores representan las probabilidades de transición cuando el procesador está en la etapa de captura. Cuando la cola de captura no está llena, un nuevo paquete entrante al sistema llega con probabilidad λ. Cuando la cola de captura está vacía, la probabilidad de que no se modifique el estado es $1-\lambda$, mientras que, cuando no está vacía, esta probabilidad es $1-\lambda-\mu_1$. Cuando la cola de captura está llena, la probabilidad de que el estado no cambie es λ.

En segundo lugar, para la acción $a = A$, el estado $s = (m, n, p)$ con $p = a = A$, y el estado $s' = (m', n', p')$,

$$p_{s,s'}^{a=A} = \begin{cases} 1-\lambda & 0 \leq m \leq M-1,\ n = 0,\ m' = m,\ n' = 0 \\ \lambda & 0 \leq m \leq M-1,\ 0 \leq n \leq N,\ m' = m+1,\ n' = n \\ 1-\lambda-\mu_2 & 0 \leq m \leq M,\ 1 \leq n \leq N,\ m' = m,\ n' = n \\ \mu_2 & 0 \leq m \leq M,\ 1 \leq n \leq N,\ m' = m,\ n' = n-1 \\ 1 & m = M,\ n = 0,\ m' = M,\ n' = 0 \\ \lambda & m = M,\ 1 \leq n \leq N,\ m' = M,\ n' = n \end{cases} \tag{3.10}$$

Al igual que en el caso de la acción $a = C$, en la Ecuación (3.10), las probabilidades asumen que

$$\lambda(1-\mu_2) \approx \lambda \tag{3.11}$$
$$(1-\lambda)\mu_2 \approx \mu_2 \tag{3.12}$$
$$(1-\lambda)(1-\mu_2) \approx 1-\lambda-\mu_2 \tag{3.13}$$

Las expresiones anteriores representan las probabilidades de transición cuando el procesador se encuentra en la etapa de análisis. Como se puede comprobar, las probabilidades de transición son análogas a las del caso con el procesador en la etapa de captura.

Por tanto, la dinámica del sistema se refleja con el proceso de estado $\dot{s}(\cdot)$ y el proceso de acción $\dot{a}(\cdot)$, que corresponden con el estado $s(t) \in \mathcal{S}$ y con la acción $a(t) \in \mathcal{A}$ en los instantes de tiempo $t \in \mathcal{T}$. Cuando el sistema está en el estado $s(t)$, la elección de una acción $a(t)$ en el comienzo de cada intervalo $t \in \mathcal{T}$, proporciona una recompensa que se contabiliza y el sistema evoluciona para pasar a otro estado en el siguiente intervalo de tiempo $t + 1$.

3.3.3. Problema de optimización

En este punto, se describe el problema de optimización con mayor detalle y se da su formulación matemática. Se denota con $\Pi_{s,a}$ al conjunto de políticas aleatorizadas y no-anticipativas (*randomized and non-anticipating policies*) disponible para el planificador. Estas políticas asumen que en el instante de tiempo $t \in \mathcal{T}$, el planificador tiene información exacta sobre la evolución del proceso de estados hasta t, es decir, conoce $s(0), s(1), \ldots, s(t)$, y también conoce todas las decisiones que se han tomado previamente, es decir, $a(0), a(1), \ldots, a(t-1)$.

El problema de optimización consiste en encontrar una política $\pi \in \Pi_{s,a}$ que maximice la esperanza de las recompensas futuras totales considerando un horizonte de tiempo, esto es,

$$\max_{\pi \in \Pi_{s,a}} \mathbb{E}_0^{\pi} \left(\sum_{t=0}^{\infty} R_{s(t)}^{a(t)} \right), \tag{P}$$

donde $\mathbb{E}_0$ es la esperanza sobre la evolución del sistema, condicionada a la información disponible en $t = 0$.

3.3.4. Solución numérica

El problema (P) es un MDP estándar, para el que se conoce que existe una política óptima que es determinista (no aleatoria), estacionaria (markoviana) e independiente del estado inicial. En particular, esto implica la existencia de una política óptima en la que, en cada instante de tiempo t, solo depende del estado del sistema $s(t)$. Se puede obtener una solución óptima markoviana del MDP resolviendo la ecuación de Bellman (véase [128][129]). Sin embargo, el hecho de que el MDP tenga como base un espacio de estados tridimensional, junto a la existencia de costes de cambio, hace que la ecuación de Bellman sea intratable desde un punto de vista analítico. Por ello, se propone resolver el MDP utilizando métodos numéricos. En concreto, se va a obtener la solución del problema siguiendo el algoritmo de iteración de valores (*value iteration algorithm*), que se describe brevemente a continuación:

Sea $\mathbb{V}_s^{\pi}$ la función valor del proceso MDP que comienza en el estado $s \in \mathcal{S}$ en el instante de tiempo $t = 0$ y siguiendo la política $\pi \in \Pi_{s,a}$, esto es,

$$\mathbb{V}_s^{\pi} = \mathbb{E}^{\pi}\left(\sum_{t=0}^{\infty} R_{s(t)}^{a(t)} | s(0) = s\right). \tag{3.14}$$

El procedimiento del algoritmo de iteración de valores sigue estos pasos:

- Se fija un valor arbitrario de inicio $\mathbb{V}_s^0$ para todo $s \in \mathcal{S}$.
- Para $i = 1, 2, \ldots, n$, calcular recursivamente:
  - $\mathbb{V}_s^i = \max_{a \in \mathcal{A}} \{R_s^a + \sum_{s' \in \mathcal{S}} p_{s,s'}^a \mathbb{V}_{s'}^{i-1}\}$
  - $a_s^i = \arg\max_{a \in \mathcal{A}} \{R_s^a + \sum_{s' \in \mathcal{S}} p_{s,s'}^a \mathbb{V}_{s'}^{i-1}\}$
- Incrementar n

Este algoritmo iterativo converge a una solución óptima del MDP, proporcionando, en el límite, dos valores importantes:

- $\lim_{i \to \infty} \mathbb{V}_s^i = \mathbb{V}_s^{\pi^*}$, y
- $\lim_{i \to \infty} a_s^i = a_s^{\pi^*}$,

donde $\mathbb{V}_s^{\pi^*}$ es la función valor, siguiendo una política óptima y comenzando por el estado s, y $a_s^{\pi^*}$ representa la acción que selecciona la política óptima en cada estado s del sistema.

Por tanto, la cuestión es obtener numéricamente la política óptima del MDP con el espacio de estados tridimensional, utilizando un conjunto de parámetros que reflejen el estudio experimental llevado a cabo sobre un sistema real de monitorización de tráfico de red. Una vez que se obtiene esta política, se podrá comparar el rendimiento del sistema real con el que de obtiene en el modelo markoviano que sigue la política óptima. Esto dará la mejora potencial del rendimiento del sistema de monitorización de tráfico cuando se utiliza una política de asignación dinámica que depende del número de paquetes en los búferes de captura y análisis.

3.4. Evaluación del modelo

En este apartado, se soluciona el problema de optimización descrito anteriormente para un determinado conjunto de parámetros. Para ello, en primer lugar, se explica cómo se obtienen los parámetros de entrada. A continuación, se describe la evaluación del modelo MDP. Para finalizar el apartado, se presentan los resultados de la simulación que sigue la política óptima obtenida del modelo MDP.

3.4.1. Estimación de parámetros

Con objeto de establecer los parámetros de entrada del modelo ($\tilde{\lambda}$, $\tilde{\mu}_1$, $\tilde{\mu}_2$, M y N), se realizan algunas pruebas en una sonda de red basada en Linux, que es el prototipo llamado Ksensor [40]. Esta sonda real está instalada dentro de una plataforma de pruebas [130], tal y como se puede ver en la Figura 3.4.

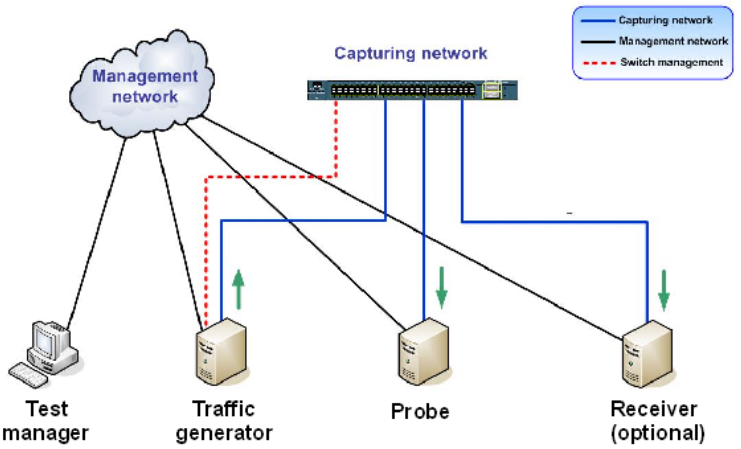


FIGURA 3.4: Plataforma de pruebas para medidas experimentales.

Dentro de la plataforma de pruebas, el generador de tráfico es responsable de enviar paquetes de datos a la red o a un potencial receptor. La sonda debe capturar y procesar el mayor número posible de paquetes de red. El equipo denominado *Test Manager*, con la ayuda de los agentes software instalados en el generador y en la sonda, mide las métricas de interés. Por tanto, el equipo gestor proporciona resultados estadísticos de las pruebas experimentales a partir de las que se estiman los valores para los parámetros de entrada del modelo matemático. Tales resultados experimentales son, por ejemplo, el número de paquetes capturados, consumos de CPU en las tareas de captura y análisis, duración de la prueba...

En lo referente a las características del hardware y software utilizados en el entorno experimental, por un lado, el generador de tráfico, el equipo gestor y el receptor están instalados sobre un mismo tipo de máquina física: un ordenador con dos CPUs Intel Xeon 5110 de 1.66 GHz, 2 GB de memoria RAM que ejecuta un Linux Debian 7. El generador de tráfico dispone de una tarjeta Endace DAG 4.3GE [11] para inyectar tráfico sintético de red. Los paquetes IP generados tienen un tamaño mínimo de 40 bytes y se utiliza una red Ethernet de 1 Gbps. Esto supone que la sonda recibirá el número máximo de paquetes posible sobre esa red. Por otro lado, la sonda está instalada sobre una máquina con dos CPUs Intel Quad Xeon 5420 de 2.5 GHz, con 4 núcleos cada una, 4 GB ded RAM y ejecuta un kernel de Linux 2.6.23 con un módulo que implementa la tarea de análisis. En los experimentos orientados hacia el modelo MDP, la sonda está configurada

para utilizar una única CPU con un solo núcleo activo. Finalmente, como muestra la Figura 3.4, todas las máquinas están conectadas al mismo conmutador o switch Ethernet de 1 Gbps.

A partir de ciertos resultados experimentales, se establece un conjunto de valores de los parámetros de entrada del modelo.

- $\tilde{\lambda}$ está relacionada con la tasa de los paquetes de red. En el escenario de pruebas, el generador de tráfico permite establecer la tasa de inyección de paquetes a la red. Se eligen 21 tasas diferentes que varían entre 50.000 y 1.500.000 paquetes por segundo (pps). Además, el equipo gestor recoge la tasa de paquetes del switch Ethernet al cual la sonda está conectado. Estos valores de tasa de tráfico de red son usados directamente como parámetro de entrada $\tilde{\lambda}$ del modelo. Ya que se mantienen los mismos valores de las pruebas experimentales, será posible comparar los resultados teóricos de la evaluación numérica del modelo con los valores experimentales. Dado que las pruebas se realizan para cada una de las tasas producidas por el generador de tráfico, hay resultados experimentales para cada una de esas tasas de tráfico de red.

- $\tilde{\mu}_1$ es la tasa de captura de paquetes. Para su estimación, se tiene en cuenta el número total de paquetes capturados por el proceso de *softirq* durante la duración de la prueba. Tanto dicho número total de paquetes capturados (al que se denotará con n_{cap}) como el tiempo dedicado por la CPU para la *softirq* (que se denotará con T_{cap}) son proporcionados por el equipo gestor. Durante la prueba, se cuentan el número de paquetes capturados por el proceso de *softirq* y el número de ciclos consumidos por la CPU en la etapa de captura. Una simple conversión de ciclos CPU a segundos facilita el valor de T_{cap}. Así, se estima $\tilde{\mu}_1$ de la siguiente manera:

$$\tilde{\mu}_1 = \frac{n_{cap}}{T_{cap}}. \tag{3.15}$$

- $\tilde{\mu}_2$ es la tasa de análisis de paquetes. Si se denota como n_{an} y T_{an} al número total de paquetes procesados en la etapa de análisis durante la prueba y el tiempo total dedicado por parte de la CPU para la etapa de análisis respectivamente, se estima el parámetro $\tilde{\mu}_2$ como sigue:

$$\tilde{\mu}_2 = \frac{n_{an}}{T_{an}}. \tag{3.16}$$

  Al igual que T_{cap}, el término T_{an} se deriva del número de ciclos consumido por la CPU en la etapa de análisis durante la prueba. Asimismo, cabe mencionar que el rendimiento de la sonda es medido bajo diferentes cargas de análisis. Se implementan tres escenarios para simular cargas de análisis bajas , medias y altas. Por ello, existen diferentes valores asignados al parámetro $\tilde{\mu}_2$, uno por cada carga de análisis. En el entorno de investigación, la implementación de estas

cargas de análisis consiste en simular su ejecución. Por ello, para cada carga de análisis, se ejecuta un bucle cuya longitud, medida en número de ciclos, varía con la intensidad de la carga de análisis que se quiere simular. Así, para la carga de análisis baja se tiene un bucle de 0 ciclos, para la carga media un bucle de 300 ciclos y para la carga alta un bucle de 1000 ciclos. De aquí en adelante, se denominará *null* al escenario de carga baja, *0.3k* al escenario de carga media y *1k* al escenario de carga alta.

- M es la capacidad del búfer de captura. En el caso de la sonda utilizada en la plataforma de pruebas, su valor típico es 200. Por ello, $M = 200$ para la evaluación numérica.

- N es la capacidad del búfer de análisis. El valor típico es 4096 en la sonda de pruebas; por tanto, $N = 4096$.

Una vez que se tienen estimados los valores para $\tilde{\lambda}$, $\tilde{\mu}_1$, $\tilde{\mu}_2$, M y N, se obtienen λ, μ_1 y μ_2 aplicando las Ecuaciones (3.1), (3.2) y (3.3) respectivamente. También se calcula la duración de cada intervalo de tiempo, t_s, aplicando (3.4). Sobre estos intervalos de tiempo, se simulará el modelo de tiempo discreto.

Otro aspecto clave del modelo es que el sistema de monitorización de tráfico consume tiempo cada vez que el procesador realiza un cambio de tarea. El procesador necesita una cantidad de tiempo no despreciable para cambiar de la etapa de captura a la de análisis, y lo mismo ocurre cuando se mueve en sentido contrario, esto es, de análisis a captura. Sea t_{sw} el tiempo de cambio de contexto (o *switching time*), su valor se deriva del escenario de pruebas, concretamente, de uno con una tasa de tráfico de paquetes alta. Bajo esta condición, se supone que no hay intervalos en los que el procesador esté libre (*idle*).

En primer lugar, se calcula el tiempo que el sistema pasa cambiando de *posición* el procesador, T_{sw}. Si la duración total de la prueba es T_{tot} (valor registrado por el equipo gestor), se tiene que

$$T_{sw} = T_{tot} - T_{cap} - T_{an}, \tag{3.17}$$

Dividiendo T_{sw} por el número de cambios (*switches*), n_{sw}, se obtiene el tiempo que el procesador necesita para hacer cada cambio:

$$t_{sw} = \frac{T_{sw}}{n_{sw}}. \tag{3.18}$$

El número de cambios se deriva del número de *softirqs* contabilizado durante una prueba. El número de *softirqs* también es registrado por el equipo gestor. Si no hay periodos libres o inactivos con tasas de datos altas, se supone que cada proceso de *softirq* es seguido por un proceso de análisis y viceversa. La CPU está dedicada prácticamente todo el tiempo para estos dos procesos, salvo el cambio de contexto entre ambos. Por tanto, puede considerarse que hay dos cambios de contexto por cada *softirq*, una en la

$\tilde{\lambda}$ (pps)	$\tilde{\mu}_1$ (pps)	$\tilde{\mu}_2$ (pps)	M	N	SC
50.000 ... ... 1.500.000	1.188.786	902.798 (*null*) 289.119 (*0.3k*) 116.755 (*1k*)	200	4096	0.4 (*null*) 0.2 (*0.3k*) 0.1 (*1k*)

Tabla 3.1: Conjunto de valores para los parámetros de entrada de la evaluación numérica

transición *softirq-análisis* y otra en la transición *análisis-softirq*. De esta forma, se determina el valor de n_{sw}.

Finalmente, se calcula el valor del coste del cambio de contexto, que permitirá definir una estructura de recompensa más significativa para el problema de optimización. Dado que se fija 1 unidad de recompensa cada vez que se completa el análisis de un paquete, parece razonable establecer el coste del cambio de contexto como la proporción de tiempo perdido durante el cambio comparado con el tiempo total que es necesario para procesar un paquete desde que llega al sistema hasta que lo abandona cuando se completa el análisis. Dicho esto, el coste del cambio de contexto se calcula de la siguiente forma:

$$SC = \frac{t_{sw}}{\frac{1}{\tilde{\mu}_1} + \frac{1}{\tilde{\mu}_2}}. \tag{3.19}$$

En el escenario con la tasa de tráfico de red más alta, se estima el tiempo del cambio de contexto y su coste asociado. Se mide experimentalmente el tiempo del cambio de contexto y da un valor $t_{sw} = 850$ ns para todos los escenarios, mientras que el coste del cambio de contexto depende de los valores de $\tilde{\mu}_1$ y $\tilde{\mu}_2$, según indica la Ecuación (3.19).

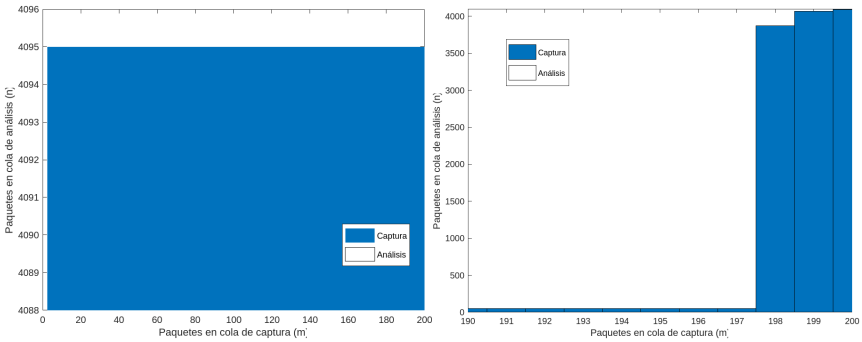
La Tabla 3.1 muestra los valores de los parámetros de entrada estimados a partir del escenario de pruebas. Con objeto de no hacer demasiado grande el tamaño de la tabla, no se han escrito los 21 valores asignados a $\tilde{\lambda}$, sino que se han puesto el valor más alto y el más bajo. La tabla también expone el valor medio de $\tilde{\mu}_1$ y tres valores medios de $\tilde{\mu}_2$ que corresponden a cada una de las cargas de análisis (*null*, *0.3k* y *1k*). Tanto los valores de $\tilde{\lambda}$ como los de $\tilde{\mu}_1$ y $\tilde{\mu}_2$ vienen dados en pps. Igualmente, la tabla da los valores usados para M, N y SC. Como resultado de la combinación de todos los posibles valores tomados para $\tilde{\lambda}$, $\tilde{\mu}_1$, $\tilde{\mu}_2$, M, N y SC, puede decirse que se evalúa el modelo para 63 escenarios diferentes.

3.4.2. Estructura de la política óptima

Después de haber estimado los valores de los parámetros de entrada, en este apartado, se evalúa numéricamente el modelo MDP y se analiza la estructura de la política óptima obtenida aplicando el algoritmo de iteración de valores para cada escenario propuesto en la Tabla 3.1. Para un escenario dado, la política óptima se plasma sobre dos matrices. La primera matriz, $C = (c_{i,j})_{i=1,\dots,M,j=1,\dots,N}$, describe la acción tomada por la

política óptima cuando el procesador está asignado a la cola de captura y hay i y j paquetes encolados en las colas de captura y análisis respectivamente. La segunda matriz, $A = (a_{i,j})_{i=1,\ldots,M,j=1,\ldots,N}$, presenta la elección óptima cuando el procesador está asignado a la cola de análisis. Como se verá más adelante, las matrices C y A serán representados en diagramas bidimensionales donde se utiliza el color azul para indicar el conjunto de estados donde la política óptima consiste en asignar el procesador a la cola de captura, mientras que el área blanca indicará que, en esos estados, es óptimo dedicar el procesador a tareas de análisis. Según las simulaciones realizadas, se ve que la forma de la política óptima depende del conjunto de parámetros de cada escenario, y se concluye que la política óptima tiene una curva que establece el límite entre las dos áreas. Sin embargo, se observa que la forma de esa curva límite sigue un patrón específico que depende de la saturación del sistema. Para explicar mejor esta situación, a continuación, se dan algunos ejemplos donde se distinguen tres rangos de tasas de tráfico de red.

En primer lugar, la Figura 3.5 muestra la forma de la política óptima cuando la tasa de tráfico de red es baja, 100,000 pps, la carga de análisis es *null*, $M = 200$ y $N = 4096$. Según la política óptima, si el procesador está asignado a la cola de captura, no hay incentivo para pasarse a la etapa de análisis, a menos que el búfer de análisis esté lleno ($n = 4096$). Por otro lado, si el procesador ya está posicionado en la cola de análisis, mantenerse o cambiar a la cola de captura depende de los valores de m y n, es decir, la cantidad de paquetes esperando en los búfers, tal y como muestra la Figura 3.5(b). Para este caso concreto, la política óptima indica que, mientras el búfer de análisis no esté vacío, no hay ningún cambio de análisis a captura hasta que el búfer de captura esté casi lleno y el número de paquetes del búfer de análisis sea menor que un valor dado. Para el conjunto de parámetros de la Figura 3.5, es óptimo mantener el procesador en la cola de análisis si hay menos que 198 paquetes en la cola de captura ($m < 198$) y la cola de análisis no está vacía ($n \neq 0$). Si $m < 198$ y el búfer de análisis está completamente vacío ($n = 0$), el procesador cambia



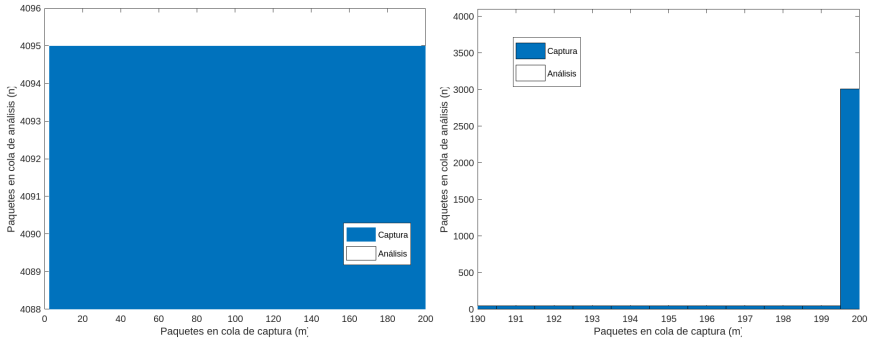
(a) Matriz C (b) Matriz A

FIGURA 3.5: Política óptima con tasa de tráfico de red de 100.000 pps y carga de análisis *null*.

a la etapa de captura. No obstante, cuando hay más de 197 paquetes en la etapa de captura ($m \geq 198$), la política óptima también depende de cuántos paquetes hay en el búfer de análisis, es decir, el procesador cambia de análisis a captura cuando el par (m, n) alcanza un determinado valor. Así, el cambio se produce si n alcanza 4092 paquetes y m es igual a 200 (búfer de captura completo) o si n disminuye a 4065 paquetes y m es igual a 199 (búfer de captura casi lleno) o si n alcanza 3875 y m es igual a 198 (búfer de captura casi lleno también). En este primer ejemplo, debido a la tasa de tráfico baja, no hay riesgo de descartar paquetes en la etapa de captura si el procesador está asignado para la tarea de análisis con menos de 198 paquetes en la cola de captura. Por ello, la política óptima es capaz de analizar el 100 % de los paquetes que llegan al sistema, minimizando, al mismo tiempo, el número total de cambios de contexto.

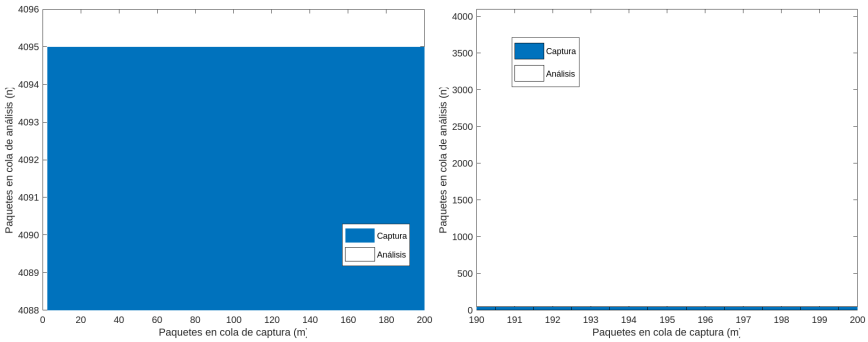
Este comportamiento de la política óptima puede interpretarse de la siguiente manera: maximizar el throughput de análisis es equivalente a minimizar la pérdida de paquetes. Por ello, siempre que la cola de análisis no esté vacía, es bueno que el procesador cambie a capturar paquetes si existe el riesgo de descartar paquetes debido a la ausencia de espacio en el búfer de captura. Por otro lado, si el número de paquetes en la cola de captura no es suficientemente grande, no hay razón para que se cambie la *posición* del procesador y, por tanto, puede mantenerse ocupado en la cola de análisis. De esta forma, se minimiza el tiempo perdido en cambios de contexto. Aun así, como este caso corresponde a una tasa de tráfico baja, el sistema dispone de suficientes recursos para transferir todo el tráfico de paquetes entrante; aunque la política, obviamente, no es perjudicial, no tiene un efecto real positivo.

En segundo lugar, la Figura 3.6 presenta la estructura de la política óptima cuando la tasa de paquetes entrante es intermedia. En concreto, es el caso de la tasa de tráfico de red de 620000 pps, carga de análisis *null*, $M = 200$ y $N = 4096$. Cuando el procesador está asignado a la cola de captura, tal y como puede verse en la matriz C de la Figura 3.6(a), resulta óptimo, otra vez, mantenerlo en esa *posición* hasta que se llene la cola



(a) Matriz C (b) Matriz A

Figura 3.6: Política óptima con tasa de tráfico de red de 620.000 pps y carga de análisis *null*.



(a) Matriz C (b) Matriz A

FIGURA 3.7: Política óptima con tasa de tráfico de red de 1.100.000 pps y carga de análisis *null*.

de análisis. Por otro lado, si el procesador está posicionado en la cola de análisis, siempre es mejor mantenerlo analizando paquetes, a menos que la cola de captura se llene ($m = 200$). Si ocurre esto, la política óptima se caracteriza por tener un valor umbral que indica si la acción óptima es cambiar la *posición* del procesador o no. Aunque la Figura 3.6 muestra el caso particular de 620.000 pps, se obtienen estructuras similares para otras tasas de tráfico intermedias (entre 600.000 y 1.000.000 pps)

A medida que la tasa de tráfico aumenta, la política definida por la matriz C permanece; también se mantiene la forma de la política de umbral que caracteriza la política dada por la matriz A para tasas de tráfico entrante intermedias, aunque el valor del umbral tiende a 0 paquetes. Precisamente, la Figura 3.7 exhibe este hecho para el escenario con una tasa de paquetes entrantes de 1.100.000 pps, una carga de análisis *null*, $M = 200$ y $N = 4096$. Una vez más, según la matriz C (véase la Figura 3.7(a)), el procesador no cambia de captura a análisis hasta que se completa el búfer de análisis ($n = 4096$). Y la matriz A de la Figura 3.7(b) indica que lo mejor es mantener el procesador analizando paquetes hasta que el búfer de análisis se vacíe por completo ($n = 0$). La razón de esto es que la tasa de tráfico entrante es tan alta que es imposible vaciar los dos búferes a la vez. Por ello, la política óptima es una política sin estados inactivos (*non-idling policy*) que minimiza el número de cambios de contexto, ya que maximiza el tiempo total que el procesador pasa capturando o analizando paquetes.

3.4.3. Rendimiento de la política óptima

Después de obtener la política óptima para cada escenario, se procede a simular la dinámica de la red de colas con la aplicación de la política óptima. El objetivo es estimar qué throughput de análisis se puede lograr si se sigue la política óptima. La Figura 3.8 muestra el resultado de la simulación. Este se puede comparar con el throughput obtenido por la sonda de tráfico real basada en Linux en la situación inicial (véase la Figura 3.2). La gráfica de política óptima indica que el rendimiento del sistema

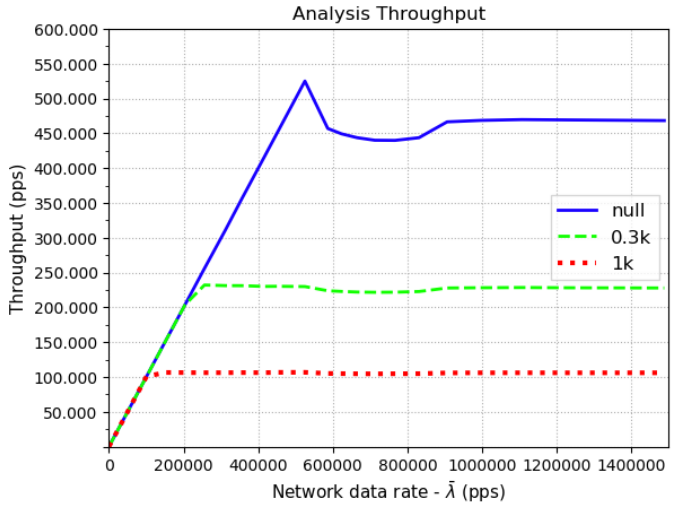


FIGURA 3.8: Rendimiento de la política óptima obtenido por simulación.

de monitorización de tráfico puede mejorar significativamente. Para ello, la asignación del procesador debe hacerse según la política óptima obtenida tras la resolución del proceso MDP derivado del modelo de la cola tándem con un solo servidor activo.

Se observa que la disminución de throughput que antes comenzaba sobre los 500.000 pps con carga de análisis *null*, sobre los 200.000 pps con carga de análisis *0.3k* y sobre los 100.000 pps con carga de análisis *1k*, desaparece. Incluso, se ve que el throughput alcanza mayores valores con tasas de tráfico bajas y medias. Además, el throuhgput de análisis se mantiene prácticamente constante para valores altos de tasa de tráfico, en contraposición con el sistema de monitorización de tráfico inicial.

Los resultados presentados en la Figura 3.8 provienen de una simulación basada en eventos discretos. Se simulan las trayectorias de los procesos de llegada de paquetes, captura y análisis son simulados por un programa que reciben los siguientes parámetros de entrada: la tasa de paquetes entrantes ($\tilde{\lambda}$), la tasa de procesado de paquetes en la etapa de captura ($\tilde{\mu}_1$), la tasa de procesado de paquetes en la etapa de análisis ($\tilde{\mu}_2$), el tamaño del búfer de captura (M), el del búfer de análisis (N), el tiempo de cambio de contexto (t_{sw}) y el contenido de la matrices C y A que, siguiendo la política óptima, apuntan la acción que se debe tomar en cada momento de decisión.

Según el modelo equivalente de tiempo discreto presentado en el apartado 3.2.2, el programa de simulación considera los intervalos de tiempo de duración t_s (definida en la Ecuación (3.4)). El programa asume que se conoce el estado actual $s = (m, n, p)$ y, más tarde, un intervalo de tiempo t_s después, el sistema va al estado $s' = (m', n', p')$ con una probabilidad de transición dada por las Ecuaciones (3.6)-(3.10). El programa genera eventos en función de las probabilidades de transición del modelo. Cuando el sistema se mueve a un estado $s' = (m', n', p')$, los valores de m' y n'

dependen del evento generado por el simulador y p' toma el valor determinado por la política óptima. Si $p = C$, el valor de la acción se toma de la matriz C y, si $p = A$, el valor de la acción se toma de la matriz A.

Otro aspecto que debe tener en cuenta el simulador es el tiempo que pasa en los cambios de contexto. Cuando el procesador está cambiando de captura a análisis o viceversa, es decir, cuando se está produciendo el cambio de contexto, el procesador no está en ninguna de las dos etapas. Por tanto, durante ese tiempo, ni se capturan ni se analizan paquetes. Sin embargo, sí es posible que nuevos paquetes lleguen al sistema. Dichos paquetes entrarán al búfer de captura o, en el peor caso, si el búfer está lleno, serán descartados y el throughput del sistema será penalizado. Como se ha mencionado previamente, el programa de simulación considera el tiempo de cambio de contexto, t_{sw}, que viene de una medida experimental. Si el tiempo del cambio de contexto, t_{sw}, es mayor que la duración del intervalo t_s, esto es, si $t_{sw} > t_s$, el programa divide el tiempo de cambio de contexto en intervalos de duración t_s y, al igual que antes, se estudia qué sucede entre el comienzo y el final del intervalo, teniendo en cuenta que el único evento posible en ese intervalo es la llegada de un paquete con probabilidad λ. Si $t_{sw} < t_s$, la probabilidad de una nueva llegada durante el tiempo del cambio de contexto es estimada con $\lambda_{switch} = \lambda \cdot \frac{t_{sw}}{t_s}$. Si $t_{sw} > t_s$ y t_{sw} no es múltiplo de t_s, se define el tiempo restante $t_{rem} = t_{sw} - kt_s$ siendo k un número entero positivo, y la probabilidad de que llegue un nuevo paquete durante el tiempo t_{rem} es igual a $\lambda_{rem} = \lambda \cdot \frac{t_{rem}}{t_s}$.

El programa de simulación controla cada variable del sistema: m, n, p, M, N. Además, diversos contadores gestionan el número de paquetes analizados, el número de paquetes entrantes, el número de intervalos de tiempo y el número de cambios de contexto. Tras simular lo que ocurre en un número de intervalos suficientemente grande, se calcula el throughput de análisis dividiendo el número de paquetes analizados por el tiempo total de la simulación. El tiempo total de la simulación es la suma del tiempo de todos los intervalos y el tiempo consumido en todos los cambios de contexto, es decir, $t_{total} = k_s \cdot t_s + k_{sw} \cdot t_{sw}$, siendo k_s el número de intervalos de tiempo y k_{sw} el número de cambios de contexto.

3.5. Validación del modelo

En el apartado anterior, los resultados de simulación han mostrado que la política obtenida a partir del modelo MDP trae consigo una mejora del throughput de análisis. Ahora, el siguiente paso será implementar la política óptima en una sonda real. Para tal fin, se utiliza la misma sonda que se ha utilizado en el estudio experimental (véase la Figura 3.4)

3.5.1. Política basada en valor umbral

Tal y como se ha explicado en el apartado 3.3, la política óptima depende del número de paquetes en la etapa de captura (m), el número de paquetes en la etapa de análisis (n) y la *posición* actual del procesador (es decir, si el procesador está ejecutando la etapa de captura o la de análisis,

en ese instante). Hacer el seguimiento de todas estas variables no es sencillo en una sonda real basada en Linux. Mientras que es factible controlar el número de paquetes en la cola de análisis, no ocurre lo mismo con el número de paquetes de la cola de captura.

Si se observa la arquitectura del subsistema de red de Linux [131] [132], el número de paquetes medido por el parámetro m corresponde con el número de paquetes almacenados en el búfer de la primera etapa, al que los paquetes llegan por transferencia DMA. Esta acción es realizada por un recurso hardware que funciona independientemente del procesador; por ello, en la sonda real utilizada en este análisis, no resulta viable tener controlado el valor real de m en todo momento. El estado de la cola puede ser conocido de una forma no demasiado precisa, puesto que puede haber diferencias entre el valor leído y el estado real de la cola. Por el contrario, el estado del búfer de análisis sí que es conocido en todo momento, ya que tanto las operaciones de escritura (por parte de la *softirq* como de lectura (por parte de la tarea de análisis) son realizadas por el procesador.

Dadas las dificultades que aparecen a la hora de monitorizar el parámetro m en el sistema real, resulta interesante analizar la estructura de la política óptima con objeto de proponer una implementación abordable. Se pretende buscar una implementación lo más cercana y consistente con la política óptima. Del estudio del apartado 3.4.2 se extrae la siguiente conclusión: cuanto mayor es la tasa de paquetes entrantes, más se parece la política óptima a una política basada en valores umbrales.

Llegados a este punto, se definen diversas políticas basadas en valor umbral para el caso en que el procesador está posicionado en la etapa de análisis:

- Política basada en un umbral T_m que depende de m. Su funcionamiento es el siguiente:

  if $m \geq T_m$ **then**
  $action \leftarrow CAPTURE$
  else
  $action \leftarrow ANALYSIS$
  end if

- Política basada en un umbral T_n que depende de n. Funciona como sigue:

  if $n \leq T_n$ **then**
  $action \leftarrow CAPTURE$
  else
  $action \leftarrow ANALYSIS$
  end if

- Política de tipo escalón que depende de m y n. Funciona como sigue:

  if $(n = 0)$ **or** $(m \geq T_m$ **and** $n \leq T_n)$ **then**
  $action \leftarrow CAPTURE$
  else
  $action \leftarrow ANALYSIS$

end if

El programa de simulación utilizado para evaluar el rendimiento de la política óptima también puede predecir resultados de políticas basadas en umbrales. Por ejemplo, la Figura 3.9 muestra el throughput de análisis que se obtendría con una política de umbral basado en m; concretamente, con un valor $T_m = 196$. Se puede ver en el gráfico que la política con T_m no aporta beneficio, sino que, al contrario, resulta perjudicial ya que, a partir de una tasa de red alrededor de 500.000 pps, el throughput es muy bajo.

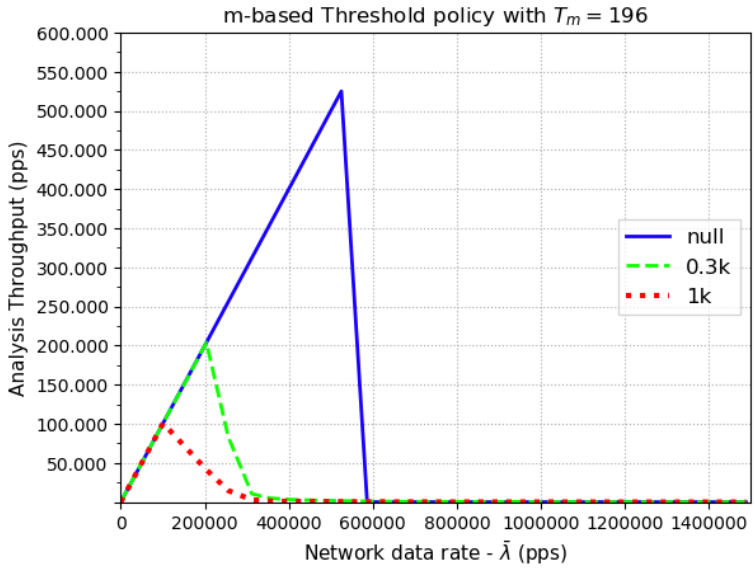


FIGURA 3.9: Simulación de rendimiento para política de umbral basa en m.

La Figura 3.10 muestra un ejemplo de política umbral basada en n con $T_n = 3000$. A diferencia de la política umbral basada en m, esta política provee una mejora de throughput considerable. Esta mejora tiene una forma similar a la de la política óptima ya que evita la caída de throughput en el rango de tasas intermedias de red y logra incrementar el throughput en el rango de las tasas más altas.

También es posible simular el comportamiento de una política con forma de escalón. La Figura 3.11 presenta un ejemplo de este tipo de política con $T_m = 198$ y $T_n = 2048$. Los resultados también son positivos, pero su implementación resulta más complicada que la de la política umbral basada en n.

Desde el punto de vista de la implementación, las políticas basadas en valores umbral son viables, en particular, aquellas cuyo valor umbral dependen del parámetro n. Si el procesador está asignado a la tarea de captura, no hay cambio de contexto hasta que la cola de análisis esté llena o hasta que la cola de captura esté vacía. Esto ocurre para todos los valores de la tasa de paquetes entrantes, por tanto, aquí, se identifica una política

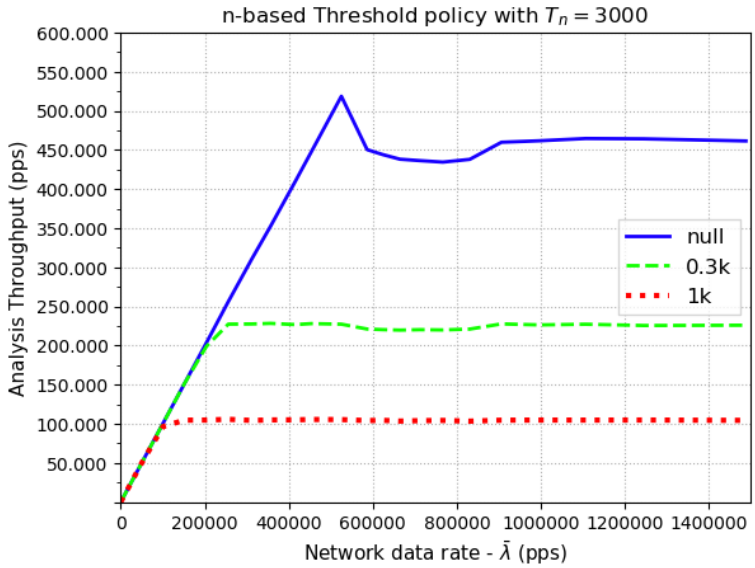


Figura 3.10: Simulación de rendimiento para política umbral basada en n.

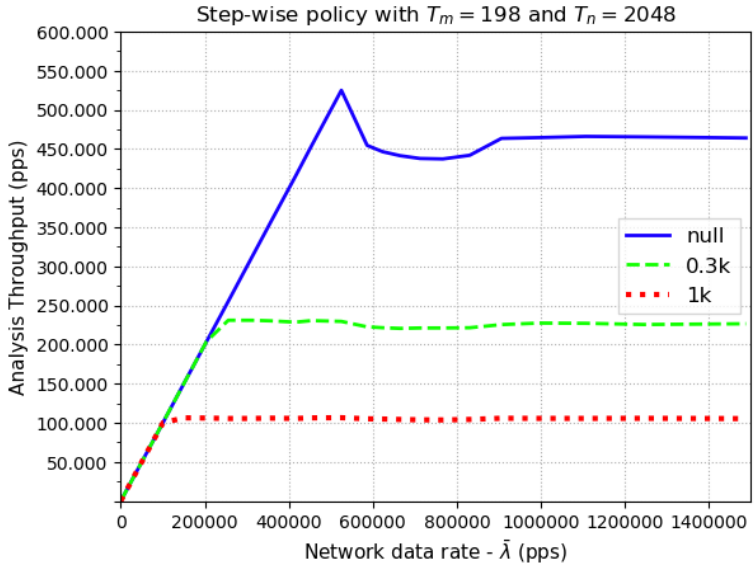


Figura 3.11: Simulación de rendimiento para política con forma de escalón.

umbral para valores $n = N$ y $m = 0$. Sin embargo, si el procesador está asignado a la tarea de análisis, el momento del cambio de contexto hacia la tarea de captura es diferente en función de la tasa de datos entrante y del número de paquetes en ambos búferes (m y n). Se observa lo siguiente,

si se distinguen tres rangos de tasas de datos entrantes:

En primer lugar, con tasas bajas de tráfico entrante, la política óptima indica que, mientras el búfer de análisis no esté vacío, no hay ningún cambio de análisis a captura hasta que el búfer de captura está casi lleno y el número de paquetes del búfer de análisis es menor que un determinado valor. Tal y como se ve en la Figura 3.5, para una tasa de paquetes entrantes de 100.000 pps y una carga de análisis *null*, la probabilidad de $m > 197$ es muy baja y, como consecuencia, es posible establecer un valor umbral T_n con $n = 0$, que es el valor común propuesto por la política óptima para $m < 197$. Sin embargo, a medida que la tasa de paquetes entrantes aumenta, también crece la probabilidad de que el búfer de captura esté lleno o casi lleno. En estos casos, el valor umbral con $n = 0$ podría no ser apropiado, ya que no evitaría el riesgo de descartar paquetes a la entrada del búfer de captura.

En segundo lugar, con tasas intermedias de paquetes entrantes, sí es necesario considerar en caso en el que el procesador está analizando paquetes, el búfer de captura está casi lleno y el procesador debe cambiar de análisis a captura con objeto de analizar el 100 % del tráfico entrante sin descartar ningún paquete a la entrada del sistema. Por ejemplo, para la tasa de paquetes entrantes de 525.000 pps y carga de análisis *null*, la política óptima indica que el procesador tiene que cambiar de análisis a captura cuando m es mayor o igual que 194; en el resto de casos, la tarea de análisis no finaliza hasta que el búfer se vacía por completo. Si $m \geq 194$, el valor de n del punto (m, n) en el que se produce el cambio de contexto difiere de un caso a otro. En concreto, se tienen los siguientes pares: $(m = 194, n = 3609)$, $(m = 195, n = 3840)$, $(m = 196, n = 3910)$, $(m = 197, n = 3951)$, $(m = 198, n = 3978)$, $(m = 199, n = 3996)$, $(m = 200, n = 4010)$. Como parece lógico, cuanto mayor es la ocupación del búfer de captura, más pronto (es decir, con un valor de n mayor) debe producirse el cambio. Sin embargo, como se ha mencionado previamente, la implementación de la política basada en los dos parámetros, m y n, es compleja. Por esta razón, se optará por fijar una política basada en umbral que solo depende de n.

Después, con tasas de tráfico intermedias más altas, por ejemplo 620.000 pps, tal y como se observa en la Figura 3.6 la política óptima se caracteriza por un valor umbral, ya que el procesador solo cambia de análisis a captura cuando el búfer está completo. Por tanto, el valor umbral es $m = M$, pero, al igual que antes, dada la dificultad para monitorizar la ocupación del búfer de captura, esto se puede trasladar a un valor umbral basado en n.

Finalmente, con las tasas de datos más altas, la política óptima es una política sin periodos inactivos (*non-idling policy*) que puede ser expresada completamente en términos de un valor umbral basado en n. Por ejemplo, como se puede ver en la Figura 3.7, para la tasa de paquetes entrantes 1.100.000 pps, el procesador cambia de análisis a captura cuando el búfer de análisis queda vacío. Así, en este caso, el valor umbral es $T_n = 0$ para todos los valores de m, siendo $0 \leq m \leq M$.

3.5.2. Comparativa entre throughput óptimo y real

Como se acaba de ver, se introduce un mecanismo de control en la sonda real. Funciona de la siguiente manera: cada vez que se introduce un nuevo paquete en la cola de análisis (el búfer de la aplicación de monitorización), el sistema supervisa el número de paquetes en dicha cola y no cambia a la tarea de analizar paquetes siempre que la cola de análisis no esté llena ($n < N$) y que la cola de captura no esté vacía ($m > 0$). Este comportamiento sigue exactamente el resultado de la política óptima de la matriz C. Cuando la cola de análisis se llena ($n = N$), esto significa que se ha alcanzado el umbral de captura y el sistema debe parar la ejecución de la tarea de captura para pasar a ejecutar la tarea de análisis. Para llevar a cabo esta implementación en el sistema real, teniendo en cuenta el procedimiento New API (NAPI) de Linux [132], es preciso mantener desactivadas las *hardirqs* y detener la ejecución de la función *poll()* del proceso de *softirq*. Hecho eso, en ese momento, la instancia de análisis empezará a procesar los paquetes de su búfer, provocando la salida de paquetes del sistema. Este procesamiento tendrá lugar hasta que se alcance el umbral de análisis, T_n, que se ha estimado previamente. Es en ese momento cuando se habilita de nuevo la tarea de captura.

Se puede afirmar que, para implementar este mecanismo de control, se deben establecer dos umbrales: T_{max} o umbral máximo cuyo valor depende del tamaño de la cola de análisis, $T_{max} = N$, y el umbral mínimo con valor $T_{min} = T_n$.

La arquitectura de pruebas utilizada (véase la Figura 3.4) permite la implementación de esta política umbral con $T_{max} = N$ y $T_{min} = T_n$, ya que la tasa de datos de la red es controlable a través del generador de tráfico. La Figura 3.12 presenta el throughput de análisis de la sonda real con el mecanismo de control implementado para diferentes tasas de paquetes entrantes y diferentes cargas de análisis. Se observa que el resultado obtenido es muy próximo al resultado de rendimiento obtenido mediante simulación de la Figura 3.8.

La Figura 3.13 expone la diferencia entre la política óptima y la implementada en términos de eficiencia de throughput, suponiendo que la política óptima logra un valor de 1,0 (100 %). El resultado indica que, aunque la política implementada no coincide con la óptima, la diferencia de resultado entre ambas no es mayor que un 2 % en todo el rango de valores de las tasas de paquetes entrantes. La Figura 3.13 también muestra resultados de simulación de las políticas con forma de escalón basadas en T_n y T_m. En el caso de las políticas basadas en T_n, ya que la sonda real permite su implementación, el resultado de simulación y el de la sonda real son muy próximos. Por tanto, se puede concluir que la implementación real es aceptable y que la herramienta de simulación reproduce correctamente el comportamiento del sistema real

Otro aspecto a considerar en la implementación de la sonda real es la variabilidad del umbral T_n con respecto a la tasa de datos entrante. Como se ha comentado previamente, en el escenario experimental con una fuente de tráfico sintético, es viable realizar pruebas variando el parámetro T_n en función de una tasa *controlada* de datos entrantes. Sin embargo, en un

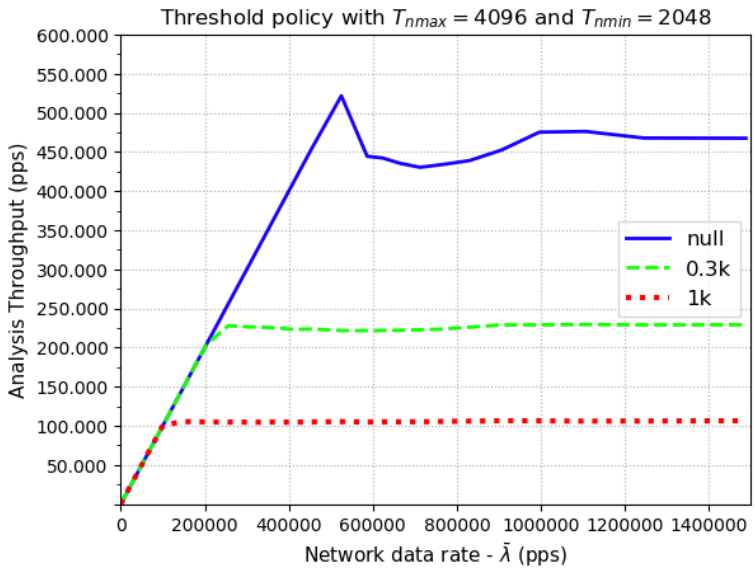


FIGURA 3.12: Rendimiento de la sonda real con el mecanismo de control aplicado.

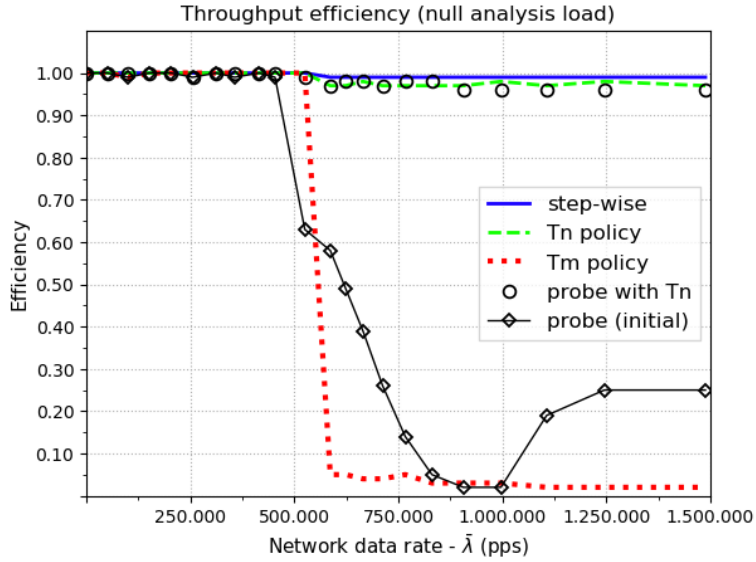


FIGURA 3.13: Comparación de eficiencias de throughput.

entorno de red, puede ser complejo controlar la velocidad del flujo de datos y adaptar la política umbral basada en n simultáneamente. Por esta razón, se decide implementar una política umbral con el mismo valor de T_n

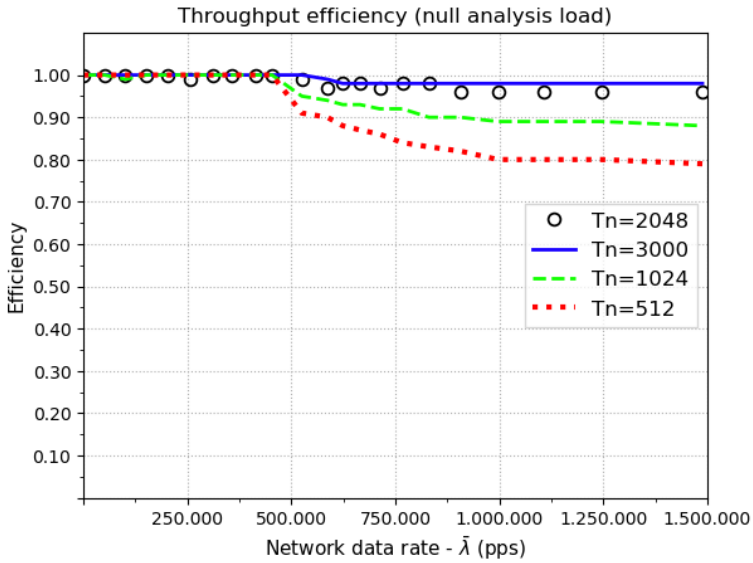


FIGURA 3.14: Comparición de rendimiento de la sonda real con diferentes políticas umbrales basadas en n.

para todas las tasas de paquetes entrantes y todas las cargas de análisis. Así, se hacen pruebas con distintos valores de T_n (512, 1024, 2048 y 3000). La Figura 3.14 presenta resultados de esas pruebas con una comparación en términos de eficiencia de throughput con respecto a la política óptima. Resulta significativa la mejora de throughput que se logra cuando el mecanismo de control actúa con tasas de entradas que sobrecargan el sistema. Obviamente, la política implementada en el sistema real no alcanza a la política óptima, pero el resultado obtenido es muy aceptable. Para algunas tasas de datos, la mejora de rendimiento obtenida es de hasta un 80 %. Además, destaca la fiabilidad de la herramienta de simulación desarrollada para los casos en que ha sido posible realizar una implementación real.

3.6. Representación del modelo con red de Petri

El modelo basado en MDP que se ha desarrollado en este capítulo también se ha representado mediante una red estocástica de Petri y, posterioremente, se ha evaluado mediante simulación. Este apartado describe esa otra forma de modelado.

3.6.1. Introducción a las redes de Petri

Antes de presentar el modelo es necesario introducir brevemente los fundamentos de las redes de Petri [133]. Esta técnica de modelado permite describir de una manera formal sistemas discretos distribuidos, especialmente aquellos cuya dinámica se caracteriza por tener conflictos de concurrencia y sincronización [134]. Las redes de Petri combinan teoría matemática con una representación gráfica de la dinámica del comportamiento del sistema. Por ello, permiten realizar el análisis y el modelado del comportamiento del sistema, a la vez que disponen de una representación gráfica que visualiza los cambios de estado del sistema modelado.

Las redes de Petri tienen dos tipos de nodos: los lugares (*places*) y las transiciones. Los lugares se dibujan mediante círculos y describen posibles estados del sistema. Las transiciones se dibujan mediante barras o cajas y se usan para describir eventos que podrían modificar el estado del sistema. Los lugares y las transiciones se conectan por arcos. Estos especifican la relación entre estados y eventos indicando, por un lado, el estado en el que el evento puede ocurrir y, por otro lado, las transformaciones inducidas por el evento en el estado del sistema. Los arcos pueden ser de tres tipos:

- Arcos de entrada: arcos acabados en flecha que van de los lugares a las transiciones.
- Arcos de salida: arcos acabados en flecha que van de las transciones a los lugares.
- Arcos inhibidores: arcos acabados en círculo que van de lugares a transiciones.

Otro de los elementos de las redes de Petri son los testigos o marcas. Estos se sitúan en los lugares y especifican el estado de la red de Petri. Se representan mediante puntos. Si un lugar describe una condición y puede contener un testigo o ninguno, la condición es verdadera si hay un testigo y falsa si no lo hay. Si un lugar define una situación, el número de testigos contenidos en el lugar sirve para especificar dicha situación. La Figura 3.15 muestra un ejemplo con tres lugares y una transición. *Place 1* contiene un testigo. La transición tiene un arco de entrada proveniente de *Place 1* y un arco inhibidor que viene de *Place 2*; también hay un arco de salida de la transición a *Place 3*.

La dinámica de la red de Petri depende de las reglas de disparo. Una transición puede disparar (y, por tanto, ocurre un evento) si todos los lugares conectados a la transición mediante un arco de entrada contienen al menos un testigo y si todos los lugares conectados a la transición mediante un arco inhibidor no continen ningún testigo. En ese caso, se dice que la transición está habilitada. Si se utilizan arcos con pesos, el disparo de una transición habilitada retira $w(p_i, t_k)$ testigos de cada uno de los lugares de entrada p_i a la transición t_k y $w(p_j, t_k)$ testigos son depositados en cada uno de los lugares de salida p_j de t_k. Cuando una transición dispara, el estado de la red de Petri puede cambiar. El conjunto de alcanzabilidad de una red de Petri es el conjunto de todos los estados alcanzables a partir de un estado inicial M_0 disparando cualquier secuencia de transiciones.

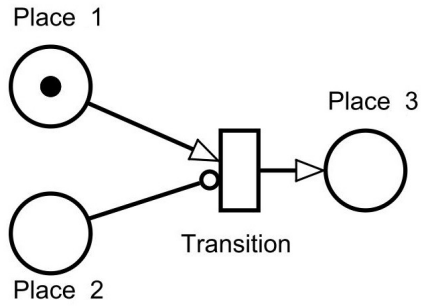


FIGURA 3.15: Ejemplo con lugares, transición, arcos y testigo.

3.6.2. Redes de Petri estocásticas generalizadas

Se va a utilizar un tipo específico de red de Petri para representar a la sonda de análisis de tráfico de red denominada red de Petri estocástica generalizada o *Generalized Stochastic Petri Nets (GSPN)*. Este tipo de redes provienen de las redes de Petri estocásticas o *Stochastic Petri Nets (GSPN)*, que introducen el concepto de tiempo. De esta forma es posible obtener parámetros de rendimiento en términos de tiempo. Las redes SPN contienen transiciones temporizadas que representan actividades que requieren un determinado tiempo para ser completadas. En las redes GSPN, no solo hay transiciones temporizadas, sino que también existen transiciones inmediatas que se utilizan para representar condiciones lógicas.

Cada transición temporizada tiene un temporizador asociado. Cuando la transición temporizada se habilita, el temporizador arranca con su valor inicial y va decrementando hasta llegar a cero. Es en ese instante cuando la transición dispara. Se considera que la actividad está en progreso durante el tiempo en que la transición está habilitada.

Por su parte, las transiciones inmediatas disparan tan pronto como se habilitan (tienen un retardo nulo). Si varias transiciones inmediatas se habilitan y pertenecen al mismo ámbito, surge un conflicto. Para resolver esto, es necesario una métrica para identificar qué transición producirá la modificación de la marca. Por ello, las redes GSPN asocian pesos a las transiciones inmediatas. Dado que estas últimas se utilizan principalmente en bifurcaciones, su función es la de determinar la probabilidad de tomar una determinada rama de ejecución u otra. Dadas dos transiciones inmediatas *I1* e *I2* en paralelo (condición de bifurcación) caracterizadas por sus pesos w_1 y w_2, la probabilidad de seguir por la rama asociada a la transición I1 será $w_1/(w_1 + w_2)$, mientras que la probabilidad de seguir por la rama de I2 será $w_2/(w_1 + w_2)$.

En la representación gráfica del modelo GSPN que se presenta a continuación, las transiciones temporizadas aparecerán como cajas rectangulares blancas, mientas que las transiciones inmediatas se dibujarán como barras negras.

3.6.3. Red de Petri para la sonda de análisis

Una vez explicados los fundamentos básicos para entender una red GSPN, a continuación se muestra el modelo basado en ese tipo de red de Petri que representa a la sonda de análisis de tráfico de red. Al igual que en el modelo basado en el proceso MDP, se pretende representar fielmente a la sonda de análisis y estimar su rendimiento en términos de throughput. La Figura 3.16 muestra la red GSPN que representa a la sonda que captura y analiza paquetes de la red. Hay que recordar que solo hay un procesador y que no es posible ejecutar las dos tareas (captura y análisis) simultáneamente. La representación gráfica permite reproducir el comportamiento dinámico del sistema.

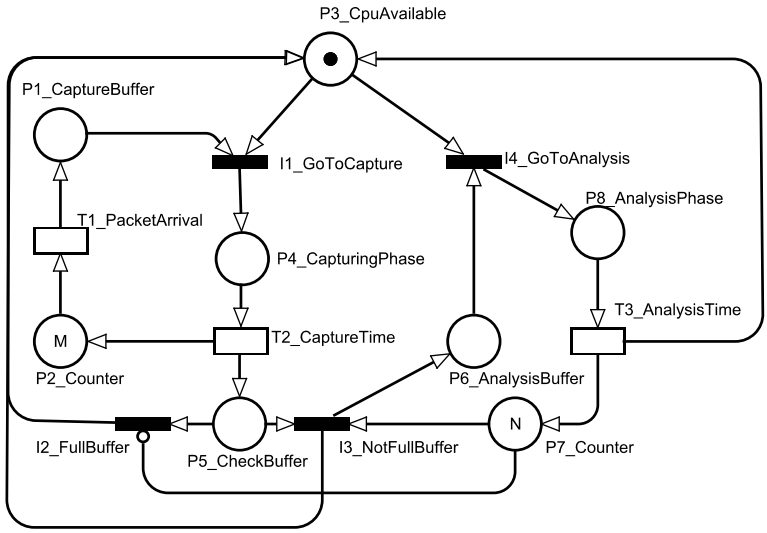


Figura 3.16: Red de Petri para la sonda con un procesador.

El estado inicial o *marca inicial* de la red de Petri tiene tres lugares con testigos: *P3_ CpuAvailable* con un testigo que indica que el procesador está disponible (no ocupado), *P2_ Counter* con M tokens, donde M es el tamaño del búfer (por ejemplo, M=200 paquetes), y *P7_ Counter* con N testigos, donde N es el tamaño del búfer de análisis (por ejemplo, N=4096 paquetes). Este estado de partida corresponde con la situación en que la sonda no tiene ningún paquete almacenado, ni en el búfer de captura ni el de análisis, por tanto, el único procesador que tiene la sonda no está ocupado en ninguna de las dos tareas.

Dada esta situación de partida, se distinguen tres partes dentro del modelo:

- Elementos que representan la llegada de paquetes al sistema.
- Elementos que representan a la fase de captura de paquetes.
- Elementos que representan la fase de análisis.

3.6.3.1. Llegada de paquetes al sistema

En la Figura 3.16, esta parte del modelo contiene los siguientes lugares y transiciones: *T1_ PacketArrival*, *P1_ CaptureBuffer* y *P2_ Counter*. *T1* es la transición temporizada que representa la llegada de paquetes al sistema con tasa λ (en paquetes por segundo). Siempre que haya algún testigo en el lugar *P2*, la transición *T1* estará activa y disparará cuando su temporizador asociado expire. Entonces, se moverá un testigo de *P2* a *P1*. Esto representa que un paquete proveniente de la tarjeta de red ha entrado en el búfer de captura. Por tanto, *P1* representa el búfer de la primera etapa.

Por el contrario, si no hay ningún testigo en el lugar *P2*, esta situación modela el estado de bloqueo del búfer de captura, puesto que habrá alcanzado su máxima capacidad M. En una situación de bloqueo, los nuevos paquetes entrantes son rechazados por el sistema. En el modelo, esto no implica ningún movimiento de testigo, pero es posible medir el flujo de paquetes rechazados, λP_B, siendo P_B la probabilidad de bloqueo. Esto es equivalente a la probabilidad de que no haya testigos en el lugar *P2*. Por ello, $P2$ es un contador que controla la ocupación del búfer de captura. Se inicializa con tantos testigos como el número máximo de paquetes del búfer de captura, M. Cada vez que la transición *T1* dispara, el contador $P2$ se decrementa en una unidad. Por el contrario, como se verá más adelante, cuando un paquete sale del búfer, el contador se verá incrementado en una unidad.

3.6.3.2. Etapa de captura de paquetes

Esta parte del modelo representa la ejecución de la tarea de captura en la que el procesador extrae paquetes del búfer de captura, realiza un tratamiento básico del paquete e inserta el paquete en el segundo búfer (búfer de análisis). Si el búfer de análisis está lleno, el paquete será rechazado y, por tanto, se perderá. Los lugares y transiciones relacionados con esta parte del modelo son: *I1_ GoToCapture*, *P4_ CapturingPhase*, *T2_ CaptureTime*, *P5_ CheckBuffer*. Además, participan dos lugares más, *P6_ AnalysisBuffer* y *P7_ Counter* que representan al búfer de análisis y su control respectivamente. Dado que el búfer de análisis es un recurso relacionado tanto con la etapa de captura como la de análisis, *P6* y *P7* aparecerán de nuevo en la siguiente etapa.

En el modelo de la Figura 3.16, la tarea de captura comienza cuando se dispara la transición inmediata *I1*. Esto ocurrirá si el procesador está disponible (testigo en *P3_ CpuAvailable*) y al menos hay un paquete en el búfer de captura (presencia de algún testigo en *P1*). Hay que mencionar que *I1* podría entrar en conflicto con la transición inmediata *I4_ GoToAnalysis* que, como se explicará más adelante, está relacionada con la ejecución de la tarea de análisis por parte del procesador. Esto ocurrirá si las dos transiciones inmediatas se habilitan al mismo tiempo. Esta condición de bifurcación de *P3* se resuelve con la asignación de pesos de *I1* y *I4*. Por ello, si w_1 y w_4 son los pesos de *I1* e *I4* respectivamente, la probabilidad de que el testigo sea retirado de *P3* y se dispare *I1* es

$w_1/(w_1 + w_4)$, mientras que la probabilidad de que el testigo sea retirado de *P3* y se dispare *I4* es $w_4/(w_1 + w_4)$.

El siguiente elemento de la etapa de captura es el lugar *P4*. Cuando hay un testigo en *P4*, esta situación indica que el procesador está ejecutando la tarea de captura, es decir, está procesando un paquete del búfer de captura; al mismo tiempo, el testigo que está en *P4* habilita la transición temporizada *T2*. Esta no se disparará hasta que su temporizador finalice. Por tanto, *T2* representa le tiempo requerido para realizar las operaciones de la etapa de captura. La transición *T2* tiene dos salidas: por un lado, un testigo irá a *P5*, el siguiente lugar de la etapa de captura; por otro lado, un testigo irá a *P2* para indicar que un paquete ha sido extraído del búfer de captura (*P1*) y, por tanto, hay una nueva posición libre en el búfer. Por ello, el contador *P2* se incrementa en una unidad.

Al finalizar la etapa de captura, un testigo llega al lugar *P5* y una de las transiciones inmediatas *I2* o *I3* se habilita y dispara. Solamente disparará una de las dos, porque no surge conflicto entre ellas. Se disparará *I2* si el contador *P7* está vacío. Esto quiere decir que no hay espacio en el búfer de análisis y el paquete será rechazado. La salida de *I2* lleva un testigo a *P3* y, por tanto, la finalización de la etapa de captura. Si en lugar de *I2*, se dispara *I3*, esto quiere decir que, al menos, hay un testigo en el contador *P7*, hay espacio disponible en el búfer de análisis e *I3* tiene dos testigos de salida: uno para *P6* para indicar que un paquete entra en el búfer de análisis y se decrementa el contador *P7* en una unidad; otro para *P3* para indicar que se ha completado la etapa de captura con ese paquete.

3.6.3.3. Etapa de análisis

Esta parte del modelo representa al procesamiento del paquete en la tarea de análisis. Los lugares y transiciones asociados a esta fase son: *I4_ GoToAnalysis*, *P8_ AnalysisPhase*, *T3_ AnalysisTime*, *P6_ AnalysisBuffer* y *P7_ Counter*.

Esta tarea comienza cuando se dispara *I4*. Esto ocurre si el procesador está disponible (testigo en *P3*) y hay algún paquete en el búfer de análisis (algún testigo en *P6*). Como ya se ha mencionado previamente, *I4* puede entrar en conflicto con *I1*. Esto se resuelve con la asignación de pesos w_1 and w_4. Después del disparo de *I4*, un testigo va a *P8* y permanece ahí hasta que se dispare la transición temporizada *T3*. Esta representa el tiempo que requiere el procesamiento de un paquete en la etapa de análisis. Cuando expira el temporizador asociado, se retirará un testigo de *P8* y dos testigos saldrán de *T3*: uno para *P7* con objeto de incrementar este contador en una unidad; otro para *P3* para indicar que se ha completado la etapa de análisis.

Como se ha mencionado previamente, *P6* representa al búfer de análisis y, en este lugar habrá tantos testigos como paquetes almacenados en dicho búfer. Y *P7* controla el número de posiciones disponibles en el búfer finito de tamaño N.

3.6.4. Simulación del modelo de red de Petri

En este apartado se muestra el resultado de la simulación de la red GSPN. Cuando la red de Petri tiene un espacio de estados muy grande, es inviable aplicar métodos analíticos o numéricos y la simulación es la solución más común. Hay muchas herramientas software para simular redes de Petri [135][136][137]. Para este caso se ha utilizado la herramienta de simulación denominada TimeNET [138]. Este simulador admite transiciones temporizadas con distribuciones exponenciales, deterministas y generales.

Antes de lanzar la simulación, es necesario introducir los parámetros de entrada del modelo que son los siguientes: los valores de los temporizadores asociados a las transiciones T1, T2 y T3, los pesos w_1 y w_4 de las transiciones inmediatas I1 e I4 y los tamaños de los búferes M y N. No obstante, la mayoría de ellos coinciden con los del modelo basado en MDP y sus valores se establecen de la misma forma que se hace en el apartado 3.4.1. Así, el valor del temporizador de T1 coincide con $1/\tilde{\lambda}$, el de T2 con $1/\tilde{\mu}_1$ y el de T3 con $1/\tilde{\mu}_2$. Los tamaños de búfer son $M = 200$ y $N = 4096$.

En lo referente a los pesos de las transiciones inmediatas, el valor de w_1 se establece a partir de la medida experimental del uso de CPU en la etapa de captura de la sonda real. Una vez fijado w_1, el valor de w_4 es inmediato, $w_4 = 1 - w_1$.

Después de establecer los valores de los parámetros de entrada, se ejecuta el simulador TimeNet configurado con un nivel de confianza del 95 %, un error relativo máximo del 5 % y una precisión que permite una diferencia máxima de un 1 % para las probabilidades próximas a 0.0 y 1.0. El simulador permite obtener los siguientes resultados del modelo: throughput de análisis (X_a), throughput de captura (X_c), probabilidad de bloqueo de captura (P_{Bc}) y de análisis (P_{Ba}) , así como el retardo medio por paquete ($\overline{T}_p$), aplicando las siguientes fórmulas:

$$\begin{aligned} X_a &= Prob\{P8\_AnalysisPhase > 0\}/T3 \\ X_c &= Prob\{P4\_CapturingPhase > 0\}/T2 \\ P_{Bc} &= Prob\{P2\_Counter == 0\} \\ P_{Ba} &= Prob\{P7\_Counter == 0\} \end{aligned}$$

La última expresión de los resultados es el retardo medio de un paquete ($\overline{T}_p$) que se calcula aplicando la fórmula de Little. Será la suma de los retardos de la etapa de captura y de análisis.

$$\overline{T}_p = \frac{\#(P1 + P4)}{X_c} + \frac{\#(P6 + P8)}{X_a}$$

donde $\#(P1 + P4)$ y $\#(P6 + P8)$ son datos que proporciona el simulador, en concreto, la suma del número medio de testigos en los lugares $P1$ y $P4$ (es decir, el número medio de paquetes en la etapa de captura) y la suma del número medio de testigos en los lugares $P6$ y $P8$ (es decir, el número medio de paquetes en la etapa de análisis).

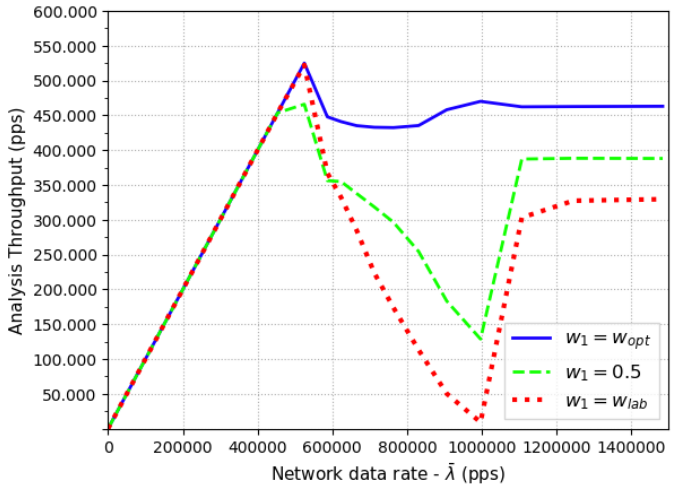


FIGURA 3.17: Resultados de throughput de la simulación de la red de Petri.

La Figura 3.17 muestra el throughput obtenido por simulación para tres situaciones diferentes, cada una de ellas con una caracterización distinta de w_1 (peso de la transición inmediata I1). La gráfica de $w_1 = w_{lab}$ corresponde con el caso en el que se introducen los valores experimentales medidos en laboratorio sobre la sonda real. El resultado es que se obtiene un rendimiento similar al que se tenía en la situación inicial del problema de optimización planteado en este capítulo. El caso $w_1 = 0{,}5$, es el resultado de una simulación donde, para todas las tasas de entrada $\tilde{\lambda}$, se mantiene el mismo valor de w_1. Finalmente, el throughput mostrado para $w_1 = w_{opt}$ es, en realidad, el resultado de un conjunto de simulaciones que, variando el valor de w_1 para cada tasa de entrada $\tilde{\lambda}$, busca con qué w_1 se alcanza el máximo de throughput. El resultado obtenido es satisfactorio, ya que es similar al del rendimiento de la política óptima del apartado 3.4.3.

3.7. Conclusiones

En este capítulo se ha presentado un modelo basado en un proceso MDP con objeto de:

- Representar el comportamiento de un sistema de monitorización de tráfico de red instalado sobre un hardware de propósito general.
- Mejorar el rendimiento de dicho sistema tomando como referencia el throughput de análisis.

Para ello, en primer lugar se ha identificado el problema de rendimiento observado en una sonda de trafico de red real. En una evaluación de rendimiento empírica preliminar (véase la Figure 3.2) se observó que, para

diferentes cargas de análisis, existe una pérdida de rendimiento significativa.

A continuación, se ha propuesto un modelo que consiste en una cola tándem de dos etapas con un único servidor activo. En este entorno, se ha formulado un proceso MDP tridimensional, con objeto de determinar la *posición* del servidor activo en cada intervalo de tiempo que considera el modelo, para que se mejore el throughput del sistema de colas. Como resultado del proceso MDP se obtiene una política óptima cuya estructura se ha analizado para un amplio rango de escenarios.

También se ha considerado la implementación de los resultados teóricos sobre una sonda real. Se ha simulado el resultado de la política óptima utilizando parámetros del sistema de la sonda real y se observa una mejora significativa del throughput. Asimismo, se ha investigado sobre la implementación más idónea en el sistema real para alcanzar o, al menos, acercarse al nivel de mejora obtenido con la teórica política óptima. Se ha concluido que una política basada en valores umbrales es la más adecuada ya que, aun no siendo la óptima, sí que aporta una mejora considerable. Con todo ello, se demuestra que es posible obtener una mejora del sistema que proviene del modelo matemático propuesto.

Para finalizar el capítulo, se ha hecho una representación alternativa del modelo mediante una red de Petri estocástica generalizada. Esta, por un lado, permite una representación gráfica que facilita reproducir la dinámica del sistema y, por otro lado, también posibilita realizar una evaluación de rendimiento. Mediante simulación de la red de Petri se ha calculado el throughput del sistema para diferentes casos y, entre ellos, también se ha buscado el máximo. El resultado ha sido satisfactorio, puesto que el throughtput máximo de la simulación de la red de Petri ha sido similar al obtenido con la política óptima del modelo basado en MDP.

Capítulo 4

Análisis y modelado de un sistema de captura mediante una cola con vacations

Como se ha visto en el Capítulo 2 sobre el estado del arte, los sistemas basados en Linux son ampliamente utilizados en arquitecturas de análisis de tráfico de red sobre hardware de propósito general. Los prototipos previamente desarrollados por el grupo NQaS también entran dentro de esa categoría. Siguiendo con el interés final de introducir mejoras de diseño en ese tipo de equipos, en este capítulo se aborda el modelado de un sistema de captura de paquetes basado en Linux. Es importante indicar que este capítulo se centra en la primera etapa de la sonda de tráfico de red, esto es, en la etapa de captura de paquetes. Dentro del trabajo de tesis, esta es la segunda propuesta de modelo matemático. Pretende realizar, primeramente, un análisis detallado del procedimiento que sigue el sistema de captura de paquetes de Linux y, en segundo lugar, plantear un modelo que represente dicho sistema y estimar parámetros de rendimiento, como throughput y retardo medio, de la etapa de captura. Para esto último es necesario resolver el modelo y evaluarlo numéricamente.

La estructura del capítulo es la siguiente. El apartado 4.1 describe el procedimiento que sigue el sistema de captura de paquetes de Linux, en concreto, el mecanismo denominado NAPI. A continuación, el apartado 4.2 define el concepto de *vacation* y describe las características de la propuesta de modelado de este capítulo: un sistema de cola de tamaño finito con un único servidor y con *vacations*. A continuación, se analizan y se resuelven tres casos (denominados M1, M2 y M3) del modelo genérico definido en el apartado 4.2. Así, el apartado 4.3 trata el modelo M1 para tiempos exponenciales y parámetro *budget* infinito; el apartado 4.4 se ocupa del modelo M2 para tiempos exponenciales y *budget* finito; y el apartado 4.5 desarrolla el modelo M3 para distribuciones de tiempo de tipo General

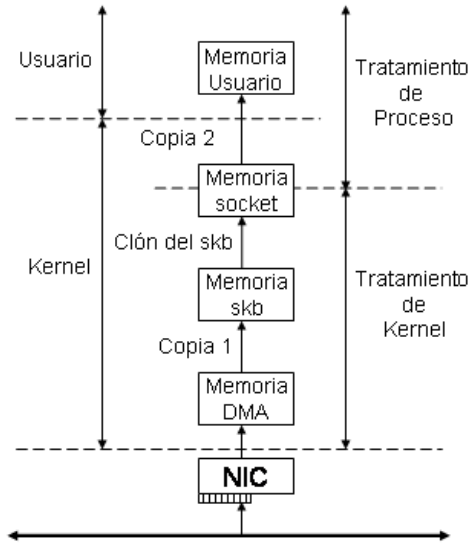


Figura 4.1: Tratamiento de kernel y de proceso.

y *budget* finito. Después de resolver los modelos mencionados (M1, M2 y M3), el apartado 4.6 indica cómo se evalúan los tres modelos y muestra los resultados obtenidos. Por último, en el apartado 4.7, se presentan las conclusiones del capítulo.

4.1. Sistema de captura de paquetes de Linux

Este primer apartado del capítulo describe el sistema de captura de paquetes que posteriormente va a ser objeto de modelado. Se ha elegido el sistema operativo Linux como base para el estudio debido a su carácter abierto y, a su vez, por ser la base de muchos sistemas de captura. Es importante incidir en que el modelo de este capítulo pretende representar únicamente el sistema de captura de paquetes de la sonda, esto es, la primera de las dos etapas que se identifican en el modelo del Capítulo 3. Por tanto, la segunda etapa de la sonda, la de análisis, queda al margen.

En la descripción del sistema, se detalla el recorrido de un paquete desde que es capturado por una tarjeta de red Ethernet hasta que llega al nivel de usuario para ser analizado por el proceso correspondiente. Para un sistema de captura de datos solo tiene relevancia el procedimiento de recepción de información del sistema de red por lo que no se va a considerar el procedimiento de emisión. Por otro lado, los paquetes capturados en modo promiscuo no atraviesan la pila TCP/IP, al no pertenecer a ninguna conexión establecida con el sistema de captura, por lo que tampoco se analizará.

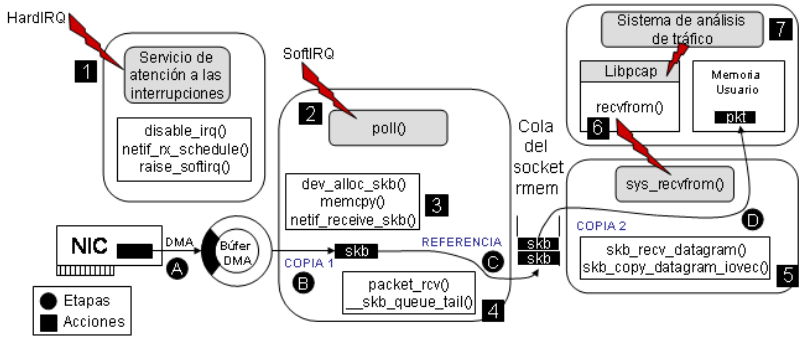


FIGURA 4.2: Diagrama del mecanismo de captura de paquetes de Linux

.

A partir del kernel 2.4.20 de Linux, la implementación de los mecanismos de captura fue modificada de forma muy similar a la descrita en [12] para hacerla más uniforme y eficiente: con la inclusión del mecanismo NAPI [139], los sistemas Linux pasaron a implementar un mecanismo de captura mixto, que es el que se explica a continuación.

La captura se divide en dos partes, atendiendo al hilo o proceso que realiza el procesamiento (ver Figura 4.1). En la primera de ellas, denominada tratamiento de kernel, se engloban todas las tareas que realiza el kernel cada vez que captura un paquete. El tratamiento de kernel se corresponde con los dos cuadros de la izquierda de la Figura 4.2. En la segunda parte de la captura, denominada tratamiento de proceso, se comentarán las acciones llevadas a cabo por el proceso de usuario para recibir los paquetes capturados por el sistema. El tratamiento de proceso se corresponde con los dos cuadros de la derecha en la Figura 4.2.

En la Figura 4.2 se distinguen acciones y etapas. Las acciones se representan mediante cuadros, mientras que las etapas mediante círculos. Las etapas son transiciones del paquete capturado a través del sistema. Por otro lado, las acciones se corresponden con todas las operaciones necesarias para que el mecanismo de captura funcione correctamente.

4.1.1. Captura y tratamiento de kernel

Una vez que la tarjeta de red ha sido inicializada correctamente, empieza a analizar las señales que recibe del medio físico. Tras detectar el preámbulo de una trama comienza su almacenamiento en una zona de memoria interna propia de la tarjeta. Después de que lleguen todos los bits de la trama, la tarjeta de red calcula el CRC del nivel Ethernet. Si no es correcto, el paquete se desecha; en caso contrario, cuando el bus I/O (PCI o similar) esté libre, el paquete es transferido a la memoria principal del sistema mediante transferencias DMA (etapa A).

El paquete se emplaza en un búfer circular (búfer DMA o *ring buffer*) en memoria de kernel, cuyo tamaño es dependiente del driver y del

hardware de la tarjeta. Una vez finalizada la transferencia DMA se genera automáticamente una interrupción hardware (*hardirq*). Hasta este momento, todas las acciones se han llevado a cabo sin que el kernel tenga constancia de ello y sin consumo de CPU.

Si la tarjeta de red introduce muchos paquetes en el búfer DMA pero el kernel no los extrae, éste finalmente se llenará. En ese caso, a menos que estuviera deshabilitada, se genera otra interrupción (distinta a la anterior) para notificarlo. A partir de ese momento, y mientras el búfer DMA siga lleno, los paquetes que se reciban se irán acumulando en la memoria interna de la tarjeta. Una vez ésta se llene, los paquetes que lleguen a continuación se desecharán.

En cualquier caso, cuando el kernel detecta la interrupción hardware (*hardirq*), y si ésta se debe a la llegada de un nuevo paquete, se ejecuta la rutina de atención a la interrupción del driver (acción 1). En esta rutina se realizan las siguientes acciones:

- Deshabilitar las interrupciones hardware de la tarjeta de red. Desde ese momento la tarjeta de red no podrá notificar si han llegado más paquetes o si el búfer DMA se ha llenado.
- Planificar la ejecución de una *softirq* que procese el paquete. Esto se hace llamando a la función *netif_rx_schedule()*.
- Confirmar a la tarjeta la correcta recepción de la interrupción.

La rutina de atención debe ser lo más corta posible para no ralentizar en exceso la tarea que se estuviera ejecutando en la CPU antes de la interrupción. Por este motivo, en la *hardirq* únicamente se realizan las acciones críticas que no se pueden retrasar (*top half*), dejando el procesamiento del paquete para más tarde, en la *softirq* planificada (*bottom half*).

El deshabilitar las interrupciones mejora el rendimiento al generarse una sola interrupción por cada grupo de paquetes capturados. Este mecanismo es conocido como coalescencia de interrupciones [12]. A pesar de que el rendimiento general del sistema aumenta, su utilización también acarrea una serie de inconvenientes.

La primera acción sobre los paquetes capturados se producirá cuando la *softirq* sea invocada (acciones 2, 3 y 4). El objetivo de la *softirq* es trasladar los paquetes desde la zona de memoria donde el DMA ha depositado los paquetes hasta la cola del socket que esté esperando los datos. Para ello, tras reservar el espacio necesario en área de kernel, copia el contenido del paquete desde el búfer DMA (etapa B). Ésta es la primera copia del contenido del paquete realizada por la CPU. Seguidamente, se indica al controlador de DMA que ese paquete ya ha sido copiado y que el espacio del búfer DMA utilizado puede ser sobrescrito.

A continuación, dentro de la misma *softirq*, se encola una referencia del paquete en la cola del socket (etapa C). Es importante destacar que no se copia todo el paquete sino únicamente una estructura de metadatos conteniendo información asociada al mismo y una referencia a su posición (que se denominará clon).

Una vez guardada la referencia al paquete en la cola del socket, la *softirq* volverá a por otro paquete (polling) y realizará las mismas acciones descritas anteriormente (etapas B y C, acciones 3 y 4). Esto se repetirá hasta que no haya más paquetes en el área de DMA utilizada por la tarjeta de red o hasta que complete un cupo máximo establecido o *weight* (en general, 64 paquetes). Por tanto, cuando la tasa de llegadas sea muy alta, la *softirq* siempre tendrá un paquete disponible para tratar por lo que normalmente finalizará tras completar su cupo máximo.

Cabe mencionar que, aunque una *softirq* no puede ser interrumpida por otra, puede haber varias ejecutándose de forma simultánea en múltiples CPUs (con las interrupciones hardware habilitadas), cada una en aquella CPU que la planificó. Existe un número limitado y restringido de tipos de *softirqs*, cada cual asociado a una tarea o acción que tiene que realizar. En la recepción de paquetes, el tipo de *softirq* que se planifica es *NET_RX_SOFTIRQ* y la acción asociada es la función *net_rx_action()*.

Finalmente, antes de terminar la rutina de atención a la *softirq*, es necesario que se vuelvan a habilitar las *hardirq*, de manera que pueda comenzar de nuevo todo el procedimiento de captura.

4.1.2. Tratamiento de proceso

Todas las acciones que se describen en esta sección se realizan en el contexto de un proceso, pero no se ejecutan exclusivamente en el entorno de nivel de usuario, sino que parte de ellas se ejecutan en el entorno de nivel de kernel (ver Figura 4.1). El tratamiento de proceso se corresponde con la actividad que se desarrolla en el kernel para atender a las demandas específicas de un proceso de usuario invocadas a través de llamadas al sistema en la interfaz socket.

La cola del socket es el punto de unión entre el tratamiento de kernel y el tratamiento de proceso. En un sistema de análisis de tráfico, el proceso de usuario es el que tiene que tomar la iniciativa y hacer una petición de los paquetes de red. Para ello se abre un socket en modo promiscuo y se ejecuta recvfrom (acciones 6 y 7). Con ello se dispara la llamada al sistema sys_recvfrom (acción 5). Esta función comprueba si hay algún paquete en la cola del socket, quedándose dormida (a la espera) en caso contrario.

Cuando el sistema operativo haya recibido un paquete y lo haya encolado, siguiendo el procedimiento descrito previamente, el proceso de usuario será despertado. A continuación, desencolará la referencia al paquete con su información asociada (clon) y se copiará el contenido del paquete de la memoria del kernel a la memoria de usuario (etapa D). Ésta es la segunda copia del contenido del paquete durante su recorrido. A modo de resumen, en la etapa A se trasladan los paquetes de la memoria de la tarjeta de red a la zona de memoria DMA. En la etapa B, se copia el paquete de la zona DMA a otra zona de memoria del kernel y se crea el socket buffer (skb). A continuación, en la etapa C se encola una referencia a ese skb en la cola del socket. Por fin, en la etapa D, se copia el paquete desde la cola del socket a la memoria de usuario.

4.2. Modelo de cola finita con vacations

El sistema de captura de paquetes descrito en el apartado anterior va a ser modelado. En concreto, el modelo deberá representar el procedimiento que el kernel del sistema operativo Linux lleva a cabo mediante la ejecución de la *hardirq* y la *softirq*. Para ello se propone un modelo de una cola de tamaño finito con un único servidor y con *vacations* o vacaciones[1]. Antes de pasar a describir el modelo, es necesario explicar qué se entiende por *vacation*.

4.2.1. Concepto de vacation

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 variadas razones [140]. Para analizar estos sistemas, se introduce el estado de *vacation*, que representa el periodo de la ausencia temporal del servidor. Por tanto, en este tipo de sistemas de colas con *vacations*, después de cada periodo activo o *busy period* 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. Este tiempo, en el que el servidor realiza dichas tareas adicionales, se denomina tiempo de *vacation* [141].

A menudo, los modelos de colas con *vacations* son adecuados para analizar planificaciones cíclicas en las que un procesador debe atender a diferentes colas de tareas [142]. El tiempo de *vacation* del servidor es equivalente al tiempo que el servidor pasa sirviendo a las otras colas, más alguna sobrecarga (*overhead*) cíclica por el cambio de tarea. En este contexto, los retardos y las pérdidas de una cola dependen en gran medida de las características que tenga ese tiempo de *vacation* del servidor.

4.2.2. Descripción del modelo

Para el modelo de una cola de tamaño finito con un único servidor y con *vacations* se tienen las suposiciones que se exponen a continuación.

La llegada de paquetes al sistema de captura sigue un proceso de Poisson con tasa λ. Este tráfico entrante representa la llegada de paquetes, vía DMA, al búfer circular. Con objeto de que la solución analítica no se vuelva inmanejable, se asume el comportamiento tipo Poisson de este proceso de llegadas. Estos paquetes son los que provienen de la tarjeta de red y son transferidos vía DMA a un área de memoria de tamaño finito.

El búfer circular se representa por una cola de espera finita de tamaño N (número máximo de paquetes que admite el búfer). En el sistema real, N depende del driver de la tarjeta de red. Si el búfer está lleno y llegan nuevos paquetes al sistema, estos serán rechazados.

[1] A partir de ahora, se utilizará el término *vacation* para expresar este estado del servidor

Dado que desde el punto de vista matemático, el caso más interesante es el de recursos limitados, se supone que solamente se dispone de un procesador o CPU, así como una única tarjeta de red para llevar a cabo la tarea de captura. Los consumos de CPU están relacionados con las capacidades de servicio de la cola, pero se tiene en cuenta que hay dos tipos de procesamiento: el asociado a la *hardirq* y el asociado a la *softirq*. Por ello, se define un tiempo de servicio de *hardirq* (τ_S), es decir, el tiempo requerido para ejecutar la Interruption Service Routine (ISR), con su tasa asociada μ_H que indica el número medio de *hardirqs* tratadas por segundo; también se tiene un tiempo medio de servicio por paquete dentro de una *softirq* (τ_S), esto es, el tiempo medio requerido para extraer cada paquete del búfer de captura, con una tasa asociada μ_S que indica el número medio de paquetes por segundo tratados dentro de una *softirq*. Además, dado que se identifican tiempos de *vacation*, tiempos en los que la CPU está inactiva o manejando procesos distintos al de la captura de paquetes, se denota como τ_V a dicho tiempo de *vacation* y su tasa asociada como μ_V (número medio de *vacations* por segundo).

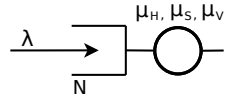


FIGURA 4.3: Cola finita para representar los procesados de hardirq, de softirq y las vacations.

Sea n la variable que indica el número de paquetes del sistema en régimen estacionario ($0 \leq n \leq N$). Sea m la variable que representa el estado del servidor donde $m = H$ indica que la CPU está ejecutando una ISR de *hardirq*, $m = V$ indica que la CPU está en *vacation*, y un valor de m entre 1 y B indica que la CPU está ejecutando una *softirq* y está atendiendo el paquete número m dentro de la *softirq* actual. B es el valor del [budget]: el número máximo de paquetes que pueden ser servidos en una *softirq*.

La Figura 4.4 muestra el diagrama de estados general del sistema. El modelo pretende recoger todos los estados y transiciones que dan lugar a representar el comportamiento del sistema de captura de paquetes con procesamiento basado en *hardirq* y *softirq*.

Supóngase que el sistema se inicie en el estado de búfer vacío ($n = 0, m = V$). Como se puede ver en la Figura 4.4, el estado de búfer vacío pertenece al conjunto de estados de *vacation*, aunque será considerado como especial ya que siempre es un estado anterior al inicio de una *hardirq*. El sistema permanece ahí hasta que llega el primer paquete con tasa λ. Es entonces cuando el sistema se mueve a un estado de procesamiento de *hardirq*, concretamente al estado inicial de *hardirq* ($n = 1, m = H$). Durante la ejecución de la ISR de *hardirq*, pueden llegar nuevos paquetes al sistema, incrementando el valor de n. Durante el tiempo de procesamiento de *hardirq* permanece en el conjunto de estados $(n, m = H)$ donde $1 \leq n \leq N$. Los nuevos paquetes son aceptados, siempre que el búfer no esté

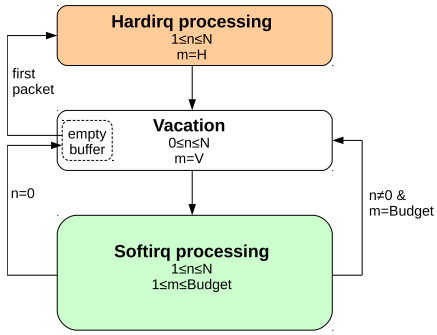


FIGURA 4.4: Diagrama de procesamientos y vacations.

completamente lleno. En caso de que el búfer no disponga de espacio libre para nuevos paquetes, estado $(n = N, m = H)$, dichos nuevos paquetes entrantes serán rechazados y el sistema permanecerá en el mismo estado.

Según el procesamiento del subsistema de red de Linux basado en NAPI, cuando finaliza la *hardirq*, ya se tiene planificada la *softirq*, pero su ejecución no es inmediata. En este caso, se dice que la *softirq* no está en contexto de *hardirq* y se ejecuta posteriormente en *ksoftirqd* [143]. Por ello, en el diagrama de la Figura 4.4, cuando termina el procesamiento de la *hardirq*, el sistema pasa a un estado de *vacation* $(n, m = V)$ donde $1 \leq n \leq N$ (obsérvese que no es posible moverse de una *hardirq* al estado de búfer vacío). Una vez más, la llegada de paquetes permanece activa independientemente del estado m. Por esta razón, durante la *vacation*, n puede seguir creciendo hasta N, como mucho.

Cuando el tiempo de *vacation* expira, tiene lugar la transición entre el final del estado de *vacation* $(n, m = V)$ y el estado inicial de *softirq* $(n, m = 1)$, donde $n < N$. El procesamiento de c es representado mediante un 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, pueden darse transiciones de estados (n, m) a $(n + 1, m)$, excepto en los estados de bloqueo (N, m). También es posible la salida de paquetes procesados, dando lugar a transiciones de estados (n, m) a $(n - 1, m + 1)$ donde $2 \leq n \leq N$ y $1 \leq m \leq B - 1$. Otro aspecto a considerar es la finalización de *softirq* que puede ser debida a las siguientes tres opciones:

- La primera es que se vacíe la cola. Según el procedimiento de Linux, cuando esto ocurre, las *hardirqs* son habilitadas de nuevo. En este caso, el sistema pasa del procesamiento *softirq* al estado de búfer vacío (transición marcada con $n = 0$ en la Figura 4.4). Esto es una transición del estado $(1, m)$, donde $1 \leq m \leq B$, al estado $(0, V)$.

- La segunda opción es alcanzar el *budget* sin que se haya vaciado completamente el búfer de paquetes. Esto corresponde a estados (n, B) donde $2 \leq n \leq N$. En este caso, Linux planifica una nueva *softirq* sin habilitar *hardirqs*. El sistema pasa del procesamiento de *softirq* a un estado normal de *vacation* (transición marcada con $n \neq 0$ y $m = B$

en la Figura 4.4). Esto es una transición del estado (n, B) al estado $(n-1, V)$ donde $2 \leq n \leq N$.

- La última opción es que haya transcurrido el límite de tiempo de 2 *jiffies*. Sin embargo, se descarta este caso para el modelo, ya que se ha verificado experimentalmente que es muy improbable.

Teniendo todo ello en cuenta, en los siguientes apartados, se propone el análisis de tres casos del modelo general presentado aquí. Se empezará con el más sencillo de los tres y se acabará con el más complejo, llamándolos de la siguiente manera:

- M1. Modelo con *budget* infinito ($B \to \infty$) y distribuciones exponenciales para los tiempos de procesamiento de *hardirq*, *softirq* y *vacations*.
- M2. Modelo con *budget* finito y distribuciones exponenciales para los tiempos de procesamiento de *hardirq*, *softirq* y *vacations*.
- M3. Modelo con *budget* finito y distribuciones generales para los tiempos de procesamiento de *hardirq*, *softirq* y *vacations*.

4.3. Modelo con vacations M1. Análisis para tiempos exponenciales y budget infinito

El primer modelo que se presenta en este capítulo, 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, 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 la tarea de capturar paquete se divide en: por un lado, una distribución exponencial de media $1/\mu_H$ para representar al tiempo de atención a la rutina de servicio de *hardirq*; por otro lado, una distribución exponenencial de media $1/\mu_S$ para los tiempos de servicio por paquete en *softirq*. Los tiempos de *vacation* siguen 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 este está completo, y serán rechazados. Por último, este modelo M1 se caracteriza por que 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.

4.3.1. Cadena de Markov asociada a la etapa de captura

Este primer modelo de cola finita con *vacations*, M1, se basa en una cadena de Markov con un espacio de estados $\mathcal{S} := \mathcal{N} \times \mathcal{M}$. Se denota al estado del sistema con $s = (n, m)$ siendo $n \in \mathcal{N} := \{0, 1, \ldots, N\}$ y $m \in \mathcal{M} := \{H, S, V\}$ $S = \{(n, m), 0 \leq n \leq N, 0 \leq m \leq 1\}$. La variable n indica el número de paquetes que hay en la etapa de captura; m identifica si el servidor está activo procesando una *hardirq* (en cuyo caso, $m = H$), activo procesando una *softirq* ($m = S$), es decir, capturando paquetes del subistema de red para un nivel superior, o si está en un periodo de *vacation* ($m = V$).

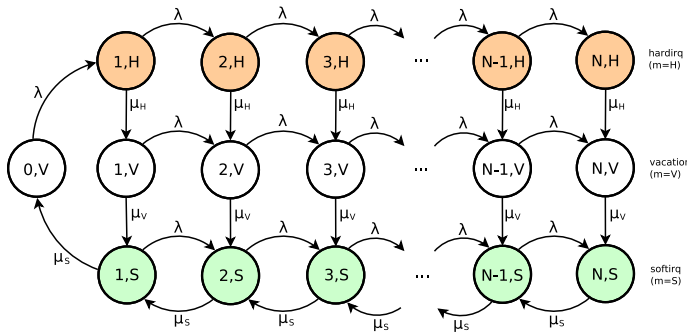


FIGURA 4.5: Diagrama de estados y transiciones del modelo M1

La Figura 4.5 muestra el diagrama de transiciones de estado en función de las tasas λ, μ_H, μ_S y μ_V. Refleja cómo procede el sistema. Primeramente, hay que indicar que, si el sistema está vacío, estado $(0, V)$, y llega un paquete nuevo, se produce el disparo de la *hardirq*, por lo que se produce la transición de $(0, V)$ a $(1, H)$ con tasa λ. Durante la ejecución de la *hardirq*, pueden llegar nuevos paquetes y, por tanto, incrementar el valor de n, permaneciendo el sistema en estados (n, H) con $1 \leq n \leq N$. Los paquetes que lleguen en el estado de bloqueo (N, H) serán rechazados.

Cuando termina la ejecución de la *hardirq*, se produce la transición del estado (n, H) al estado (n, V) con tasa μ_H. Debido a la política del planificador de tareas, el inicio del proceso de *softirq* no es inmediato, sino que se produce después de una *vacation*. Durante el periodo de *vacation*, el proceso está en los estados (n, V), donde el número de paquetes puede aumentar con tasa λ, salvo en (N, V) por haber bloqueo. También es posible terminar la *vacation*, cuando el planificador lo decida. Esto da lugar a una transición de *vacation*, estado (n, V), a *softirq*, estado (n, S) con tasa μ_V.

En los estados (n, S), el procesador está capturando paquetes. Entonces, al igual que ocurría antes, pueden entrar nuevos paquetes con tasa λ, salvo en (N, S), y salir paquetes del sistema de captura con tasa μ_S. En este modelo M1, se supone que la finalización del proceso de captura solo se da cuando se vacía el búfer: transición de $(1, S)$ a $(0, 0)$ en la Figura 4.5.

4.3.2. Ecuaciones de balance y probabilidades de estado

Sea $\pi_{n,m}$, la probabilidad del estado (n, m) en régimen estacionario. Las ecuaciones de balance [144] asociadas a los estados de la Figura 4.5 se plantean de la siguiente forma:

En primer lugar, en el estado $(0, V)$,

$$\lambda\pi_{0,V} = \mu_S\pi_{1,S} \tag{4.1}$$

En los estados $(1, m)$ donde $m \in \{H, S, V\}$,

$$(\lambda + \mu_H)\pi_{1,H} = \lambda\pi_{0,V} \tag{4.2a}$$
$$(\lambda + \mu_V)\pi_{1,V} = \mu_H\pi_{1,H} \tag{4.2b}$$
$$(\lambda + \mu_S)\pi_{1,S} = \mu_V\pi_{1,V} + \mu_S\pi_{2,S} \tag{4.2c}$$

En los estados (n, m) donde $2 \leq n \leq N-1$ and $m \in \{H, S, V\}$,

$$(\lambda + \mu_H)\pi_{n,H} = \lambda\pi_{n-1,H}, \quad 2 \leq n \leq N-1 \tag{4.3a}$$
$$(\lambda + \mu_V)\pi_{n,V} = \lambda\pi_{n-1,V} + \mu_H\pi_{n,H}, \quad 2 \leq n \leq N-1 \tag{4.3b}$$
$$(\lambda + \mu_S)\pi_{n,S} = \lambda\pi_{n-1,S} + \mu_V\pi_{n,V} + \mu_S\pi_{n+1,S}, \quad 2 \leq n \leq N-1 \tag{4.3c}$$

Finalmente, en los estados (N, m) donde $m \in \{H, S, V\}$,

$$\mu_H\pi_{N,H} = \lambda\pi_{N-1,H} \tag{4.4a}$$
$$\mu_V\pi_{N,V} = \lambda\pi_{N-1,V} + \mu_H\pi_{N,H} \tag{4.4b}$$
$$\mu_S\pi_{N,S} = \lambda\pi_{N-1,S} + \mu_V\pi_{N,V} \tag{4.4c}$$

Por otro lado, utilizando el principio de balance global [145], por el que el flujo saliente de un conjunto de estados es igual al flujo entrante a dicho conjunto

$$\lambda\left(\pi_{n,H} + \pi_{n,V} + \pi_{n,S}\right) = \mu_S\pi_{n+1,S} \qquad 0 \leq n \leq N-1 \tag{4.5}$$

Y dado que la suma de probabilidades debe ser 1.

$$\sum_{n=0}^{N}\left(\pi_{n,H} + \pi_{n,V} + \pi_{n,S}\right) = 1 \tag{4.6}$$

Tras agrupar y reescribir las Ecuaciones (4.1-4.5), se llega a una solución geométrica matricial.

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

$$\text{siendo } R = \begin{pmatrix} \frac{\lambda}{\lambda+\mu_H} & 0 & 0 \\ \frac{\lambda\mu_H}{(\lambda+\mu_V)(\lambda+\mu_H)} & \frac{\lambda}{\lambda+\mu_V} & 0 \\ \frac{\lambda}{\mu_S} & \frac{\lambda}{\mu_S} & \frac{\lambda}{\mu_S} \end{pmatrix} \tag{4.7a}$$

$$\begin{pmatrix} \pi_{N,H} \\ \pi_{N,V} \\ \pi_{N,S} \end{pmatrix} = Q \begin{pmatrix} \pi_{N-1,H} \\ \pi_{N-1,V} \\ \pi_{N-1,S} \end{pmatrix} \quad \text{siendo } Q = \begin{pmatrix} \frac{\lambda}{\mu_H} & 0 & 0 \\ \frac{\lambda}{\mu_V} & \frac{\lambda}{\mu_V} & 0 \\ \frac{\lambda}{\mu_S} & \frac{\lambda}{\mu_S} & \frac{\lambda}{\mu_S} \end{pmatrix} \tag{4.7b}$$

Imponiendo la condición de normalización de la Ecuación (4.6), se obtiene el valor de $\pi_{0,V}$.

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

$$\text{siendo } K = \begin{pmatrix} 1 & 1 & 1 \end{pmatrix} \left(\sum_{n=0}^{N-1} R^n + QR^{N-1} \right) \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \tag{4.8}$$

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

Después de calcular $\pi_{0,V}$, ya es posible determinar todas las probabilidades de estado $\pi_{n,m}$ en régimen estacionario aplicando las Ecuaciones (4.7a-4.7b).

4.3.3. Parámetros de rendimiento

Una vez que se tienen las probabilidades $\pi_{n,m}$, ya es posible calcular los parámetros de rendimiento en términos de probabilidad de bloqueo, throughput, utilización de CPU...

4.3.3.1. Probabilidad de bloqueo

La probabilidad de bloqueo, P_B, es la probabilidad de rechazar paquetes cuando el búfer está ocupado por completo. Esto ocurre en los estados (N,H), (N,V) y (N,S).

$$P_B = \pi_{N,H} + \pi_{N,V} + \pi_{N,S} \tag{4.9}$$

4.3.3.2. Throughput de captura

Se define el throughput de captura, X_C, como el número medio de paquetes capturados por segundo. Equivale a la tasa en la que la etapa de captura es capaz de transferir paquetes a su correspondiente nivel superior. cabe mencionar que la salida de paquetes de la etapa de captura tiene lugar

en los estados (n, S) donde $1 \leq n \leq N$. La expresión del throughput de captura también está directamente relacionada con la tasa de llegada de paquetes y la probabilidad de bloqueo.

$$X_C = \mu_S \sum_{n=1}^{N} \pi_{n,S} = \lambda (1 - P_B) \tag{4.10}$$

4.3.3.3. Utilización de CPU

El parámetro U_C, utilización de CPU, considera los estados en los que la CPU está dedicada a los procesamientos de *hardirq* y *softirq*.

$$U_C = \sum_{n=1}^{N} \pi_{n,H} + \sum_{n=1}^{N} \pi_{n,S} \tag{4.11}$$

4.3.3.4. Frecuencia de hardirq

El parámetro $f_{hardirq}$, frecuencia de *hardirq*, mide el número de *hardirqs* ejecutadas por segundo. La frecuencia $f_{hardirq}$ está relacionada con el ciclo compuesto por: 1) tiempo de búfer vacío; 2) tiempo de *hardirq*; 3) tiempo de *vacation* con búfer no vacío; y 4) tiempo de *softirq*. Denotando con T_{cycle} al tiempo medio de ciclo y con $T_{softirq}$ al tiempo medio de la ejecución de una *softirq*.

$$T_{cycle} = \frac{1}{\lambda} + \frac{1}{\mu_H} + \frac{1}{\mu_V} + T_{softirq} \tag{4.12}$$

Suponiendo que la llegada de paquetes sigue un proceso de Poisson, el tiempo medio de búfer vacío es igual a $1/\lambda$ y se puede obtener $f_{hardirq}$ como:

$$\begin{aligned} \frac{1}{\lambda} &= T_{cycle} \cdot \pi_{0,V} \\ f_{hardirq} &= \frac{1}{T_{cycle}} = \lambda \cdot \pi_{0,V} \end{aligned} \tag{4.13}$$

4.3.3.5. Frecuencia de softirq

El parámetro $f_{softirq}$, frecuencia de *softirq* mide el número de *softirqs* ejecutadas por segundo. En el modelo M1, hay una *softirq* por cada *hardirq*. Por tanto, ambas frecuencias, $f_{hardirq}$ and $f_{softirq}$, son iguales.

$$f_{softirq} = f_{hardirq} = \lambda \cdot \pi_{0,V} \tag{4.14}$$

4.3.3.6. Tiempo medio de softirq

Como se ha mencionado previamente, el tiempo medio de *softirq*, $T_{softirq}$, es la duración media de una *softirq*. Su expresión se obtiene de la siguiente manera:

$$T_{softirq} = T_{cycle} \cdot \sum_{n=1}^{N} \pi_{n,S} = \frac{\sum_{n=1}^{N} \pi_{n,S}}{f_{softirq}} \tag{4.15}$$

4.4. Modelo con vacations M2. Análisis para tiempos exponenciales y budget finito

El segundo modelo que se propone en este capítulo, M2, tiene como objeto representar el sistema de captura de paquetes de Linux con mayor detalle que el modelo M1, puesto que se establece un valor finito para el parámetro B. Recuérdese que B es el *budget*, el número máximo de paquetes que pueden ser procesados dentro de una *softirq*. Puede afirmarse que, mientras que M1 tiene una disciplina de servicio exhaustiva durante la *softirq* (esta no termina hasta que el búfer queda sin ningún paquete), M2 tiene una disciplina de servicio limitada durante la *softirq* (esta termina cuando se alcanza el límite establecido por B).

El resto de las suposiciones coincide con las del modelo M1. Por tanto, se mantienen las tasas λ, μ_H, μ_S and μ_V, así como el tamaño de búfer N.

4.4.1. Cadena de Markov relacionada con la etapa de captura

El análisis del modelo M2 se basa en una cadena de Markov con un espacio de estados $\mathcal{S} := \mathcal{N} \times \mathcal{M}$. Se denota al estado del sistema con $s = (n, m)$ siendo $n \in \mathcal{N} := \{0, 1, \ldots, N\}$ y $m \in \mathcal{M} := \{H, V, 1, 2, \ldots, B\}$. Como se ha explicado en el apartado 4.2, la variable n indica el número de paquetes que hay en la etapa de captura, mientras que m permite controlar qué tipo de ocupación tiene el servidor.

- $m = H$ si el servidor está procesando una *hardirq*.
- $m = V$ si el servidor está en *vacation*.
- $1 \leq m \leq B$ si el servidor está procesando una *softirq* y está siendo servido el paquete número m dentro de la *softirq* actual.

El diagrama de estados y transiciones mostrado en la Figura 4.6 refleja el procedimiento del sistema de captura de Linux. Primeramente, si el sistema está vacío, estado $(0, V)$, 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 Figura 4.6 se tiene la transición de $(0, V)$ a $(1, H)$ con tasa λ. Durante la ejecución de la *hardirq*, el sistema está en los estados (n, H). Pueden entrar nuevos paquetes con tasa λ,

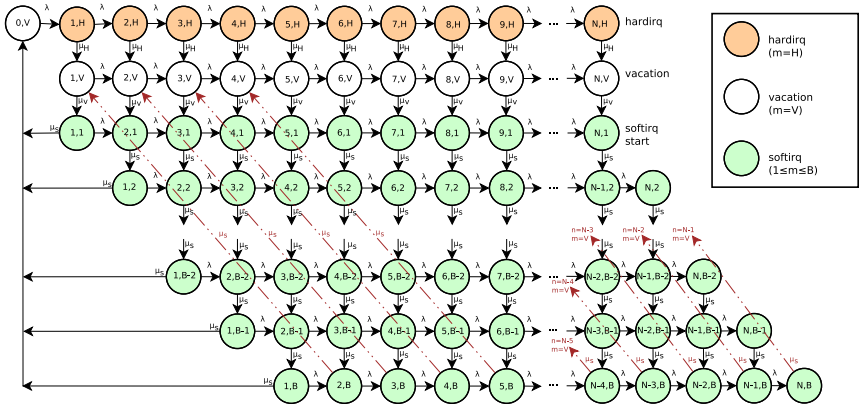


FIGURA 4.6: Diagrama de estados y transiciones del modelo M2

exceptuando en (N, H) o puede finalizarse la propia *hardirq* con tasa μ_H. Esto último implica una transición hacia un estado de *vacation* (n, V). 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, V) a un estado $(n, 1)$ con tasa μ_V.

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, como ya se ha mencionado anteriormente, se contemplan dos opciones. Una es que la cola se vacía con la salida del último paquete 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 $(0, V)$ con tasa μ_S. 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 de *vacation* $(n - 1, V)$ con tasa μ_S.

4.4.2. 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 Figura 4.6. En primer lugar, en el estado $(0, V)$,

$$\lambda \pi_{0,V} = \mu_S \sum_{m=1}^{B} \pi_{1,m} \tag{4.16}$$

En los estados $(1, m)$ donde $m \in \{H, S, 1, 2, \ldots, B\}$,

$$(\lambda + \mu_H)\,\pi_{1,H} = \lambda\pi_{0,V} \tag{4.17a}$$
$$(\lambda + \mu_V)\,\pi_{1,V} = \mu_H\pi_{1,H} + \mu_S\pi_{2,B} \tag{4.17b}$$
$$(\lambda + \mu_S)\,\pi_{1,1} = \mu_V\pi_{1,V} \tag{4.17c}$$
$$(\lambda + \mu_S)\,\pi_{1,m} = \mu_S\pi_{2,m-1} \qquad 2 \le m \le B \tag{4.17d}$$

En los estados (n, m) con $2 \le n \le N-1$ y $m \in \{H, S, 1, 2, \ldots, B\}$,

$$(\lambda + \mu_H)\,\pi_{n,H} = \lambda\pi_{n-1,H} \tag{4.18a}$$
$$(\lambda + \mu_V)\,\pi_{n,V} = \lambda\pi_{n-1,V} + \mu_H\pi_{n,H} + \mu_S\pi_{n+1,B} \tag{4.18b}$$
$$(\lambda + \mu_S)\,\pi_{n,1} = \lambda\pi_{n-1,1} + \mu_V\pi_{n,V} \tag{4.18c}$$
$$(\lambda + \mu_S)\,\pi_{n,m} = \lambda\pi_{n-1,m} + \mu_S\pi_{n+1,m-1} \quad 2 \le m \le (B-1) \tag{4.18d}$$
$$(\lambda + \mu_S)\,\pi_{n,B} = \lambda\pi_{n-1,B} + \mu_S\pi_{n+1,B-1} \tag{4.18e}$$

Finalmente, en los estados (N, m) donde $m \in \{H, S, 1, 2, \ldots, B\}$,

$$\mu_H\pi_{N,H} = \lambda\pi_{N-1,H} \tag{4.19a}$$
$$\mu_V\pi_{N,V} = \lambda\pi_{N-1,V} + \mu_H\pi_{N,H} \tag{4.19b}$$
$$\mu_S\pi_{N,1} = \lambda\pi_{N-1,1} + \mu_V\pi_{N,V} \tag{4.19c}$$
$$\mu_S\pi_{N,m} = \lambda\pi_{N-1,m} \quad 2 \le m \le (B-1) \tag{4.19d}$$
$$\mu_S\pi_{N,B} = \lambda\pi_{N-1,B} \tag{4.19e}$$

Además, se tiene la condición de normalización.

$$\sum_{n,m} \pi_{n,m} = 1 \tag{4.20}$$

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

$$\vec{\pi}_0 = \begin{pmatrix} \pi_{0,V} \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{pmatrix} \quad \vec{\pi}_n = \begin{pmatrix} \pi_{n,H} \\ \pi_{n,V} \\ \pi_{n,1} \\ \vdots \\ \pi_{n,B-1} \\ \pi_{n,B} \end{pmatrix} \quad 1 \le n \le N \tag{4.21}$$

Después de realizar el correspondiente desarrollo, se obtiene una serie de relaciones entre vectores de probabilidades en las que todos ellos quedan en función de la probabilidad $\pi_{0,V}$.

$$\begin{cases} \vec{\pi}_1 = Z_1\vec{\pi}_0 \\ \vec{\pi}_n = Z_n\vec{\pi}_{n-1} \quad 2 \le n \le N \end{cases} \tag{4.22}$$

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

$$\begin{cases} Z_N = -\lambda A^{-1} \\ Z_n = -\lambda \left(A - \lambda I + BZ_{n+1}\right)^{-1} & 2 \leq n \leq N-1 \\ Z_1 = -\left(A - \lambda I + BZ_2\right)^{-1} C \end{cases} \tag{4.23}$$

Las matrices A, B y C, que aparecen en (4.23), son cuadradas, de dimensiones $(B+2) \times (B+2)$. Contienen los siguientes elementos:

$$A = \begin{pmatrix} -\mu_H & 0 & 0 & 0 & \dots & \dots & 0 \\ \mu_H & -\mu_V & 0 & 0 & \dots & \dots & 0 \\ 0 & \mu_V & -\mu_S & 0 & \dots & \dots & \vdots \\ 0 & 0 & 0 & -\mu_S & \dots & \dots & \vdots \\ \vdots & \vdots & \vdots & 0 & \ddots & \ddots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \ddots & \ddots & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 & -\mu_S \end{pmatrix} \tag{4.24}$$

$$B = \begin{pmatrix} 0 & 0 & \dots & \dots & \dots & 0 & 0 \\ 0 & 0 & \dots & \dots & \dots & 0 & \mu_S \\ 0 & 0 & 0 & \dots & \dots & 0 & 0 \\ 0 & 0 & \mu_S & 0 & \dots & \vdots & \vdots \\ 0 & 0 & 0 & \mu_S & 0 & \vdots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \dots & \dots & 0 & \mu_S & 0 \end{pmatrix} \tag{4.25}$$

$$C = \begin{pmatrix} \lambda & 0 & \dots & \dots & 0 & 0 \\ 0 & 0 & \dots & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & & \vdots & \vdots \\ \vdots & \vdots & & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & \dots & 0 & 0 \\ 0 & 0 & \dots & \dots & 0 & 0 \end{pmatrix} \tag{4.26}$$

Sustituyendo $\pi_{n,m}$, con $(1 \leq n \leq N, -1 \leq m \leq B)$, en la condición de normalización (4.20) por las expresiones extraídas de (4.21)-(4.23), se obtiene el valor de $\pi_{0,V}$.

$$\pi_{0,V} = \frac{1}{S}; \quad S = 1 + \left[\sum_{n=1}^{N} \vec{e}_1^T \left(\prod_{i=0}^{n-1} Z_{n-i}\right) \vec{e}_2\right] \tag{4.27}$$

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

$$\begin{aligned}\vec{e}_1^T &= (1 \quad 1 \quad \ldots\ldots \quad 1)\\ \vec{e}_2^T &= (1 \quad 0 \quad \ldots\ldots \quad 0)\end{aligned} \tag{4.28}$$

Una vez que se tiene el valor de $\pi_{0,V}$, ya se pueden determinar todas las probabilidades de estado $\pi_{n,m}$, aplicando (4.22).

4.4.3. Parámetros de rendimiento

En este momento, con las probabilidades de estado en régimen estacionario, es posible estimar los parámetros de rendimiento de interés.

4.4.3.1. Probabilidad de bloqueo

La probabilidad de bloqueo (P_B) es igual a la suma de las probabilidades de los estados de bloqueo (N,m) donde $m=H$ o $m=V$ o $1\leq m\leq B$. Como ya se ha mencionado previamente, en todos estos estados de bloqueo se rechaza la entrada de nuevos paquetes al sistema.

$$P_B = \pi_{N,H} + \pi_{N,V} + \sum_{m=1}^{B} \pi_{N,m} \tag{4.29}$$

4.4.3.2. Throughput de captura

La definición de este parámetro, throughput de captura (X_C), es la misma que la dada para el modelo M1. Su expresión tiene en cuenta tanto la probabilidad de ejecutar una *softirq* como la tasa de servicio por paquete μ_S.

$$X_C = \mu_S \sum_{n=1}^{N}\sum_{m=1}^{B} \pi_{n,m} \tag{4.30}$$

4.4.3.3. Utilización de CPU

El parámetro U_C, utilización de CPU, considera los estados en los que la CPU está dedicada al procesamiento de *hardirq* y *softirq*.

$$U_C = \sum_{n=1}^{N} \pi_{n,H} + \sum_{n=1}^{N}\sum_{m=1}^{B} \pi_{n,m} \tag{4.31}$$

4.4.3.4. Frecuencia de hardirq

El parámetro $f_{hardirq}$, frecuencia de *hardirq*, mide, en media, cuántas veces por segundo se ejecuta la rutina de servicio de la *hardirq*. Tal y como se hizo en el modelo M1, se considera el tiempo medio de ciclo, T_{cycle}, para calcular $f_{hardirq}$. Aquí, el tiempo medio de ciclo consta del tiempo medio de búfer vacío, el tiempo medio de ejecución de *hardirq* y el tiempo medio

del conjunto de *vacations* y *softirqs* hasta que el búfer de paquetes se vacía completamente.

$$T_{cycle} = \frac{1}{\lambda} + \frac{1}{\mu_H} + k_s \left(\frac{1}{\mu_V} + T_{softirq} \right) \tag{4.32}$$

Se define $T_{softirq}$ como el tiempo medio de *softirq* y k_s como el número medio de *softirqs* dentro del ciclo. Puesto que se asume que la llegada de paquetes sigue un proceso de Poisson:

$$\begin{aligned} \frac{1}{\lambda} &= T_{cycle} \cdot \pi_{0,V} \\ f_{hardirq} = \frac{1}{T_{cycle}} &= \lambda \cdot \pi_{0,V} \end{aligned} \tag{4.33}$$

4.4.3.5. Frecuencia de softirq

Al igual que en el modelo M1, el parámetro $f_{softirq}$, frecuencia de *softirq*, indica el número medio de *softirqs* ejecutadas por segundo. En el modelo M2, siempre hay una *vacation* con $n > 0$ antes de cada *softirq*. Por ello, se estima $f_{softirq}$ como sigue:

$$\begin{aligned} k_s \cdot \frac{1}{\mu_V} &= T_{cycle} \sum_{n=1}^{N} \pi_{n,V} \\ f_{softirq} = \frac{k_s}{T_{cycle}} &= \mu_V \sum_{n=1}^{N} \pi_{n,V} \end{aligned} \tag{4.34}$$

4.4.3.6. Tiempo medio de softirq

Se puede obtener el tiempo medio de *softirq*, $T_{softirq}$, de la siguiente manera:

$$T_{softirq} = \frac{\sum_{n=1}^{N} \sum_{m=1}^{B} \pi_{n,m}}{\mu_V \sum_{n=1}^{N} \pi_{n,V}} \tag{4.35}$$

4.4.3.7. Número medio de paquetes de una softirq

Una vez calculado el tiempo medio de *softirq*, se estima el número medio de paquetes procesados dentro de una *softirq*, $\overline{m}_{softirq}$.

$$\overline{m}_{softirq} = \frac{T_{softirq}}{\frac{1}{\mu_S}} = \mu_S T_{softirq} = \frac{\mu_S \sum_{n=1}^{N} \sum_{m=1}^{B} \pi_{n,m}}{\mu_V \sum_{n=1}^{N} \pi_{n,V}} \tag{4.36}$$

4.5. Modelo con vacations M3. Análisis para tiempos de tipo General y budget finito

En este apartado se explica la tercera propuesta para representar la etapa de captura de paquetes de Linux. Se llamará M3. Al igual que el modelo M2, M3 también es un modelo con *vacations* y disciplina de servicio limitada por el *budget* B de *softirq*. Sin embargo, a diferencia de M1 y M2, en esta aproximación, no son exponenciales los tiempos de *hardirq*, de tratamiento de paquete en *softirq* y de *vacation*. En su lugar, todos estos tiempos siguen distribuciones de tipo General.

Para este tercer modelo, M3, se tienen las siguientes suposiciones:

- Los paquetes entrantes llegan al sistema según un proceso de Poisson de tasa λ.
- Solamente hay un único procesador para la captura de paquetes.
- El tiempo de procesado de *hardirq*, H, el tiempo de servicio por paquete en *softirq*, S, y el tiempo de *vacation*, V, son todos ellos variables independientes con funciones de distribución $H(x)$, $S(x)$ y $V(x)$ respectivamente. Sus correspondientes transformadas de Laplace-Stieltjes son $H^*(\theta)$, $S^*(\theta)$ y $V^*(\theta)$ [146].
- Hay un *budget* finito denotado como B.
- Existe un búfer de tamaño finito para almacenar paquetes antes de ser procesados por la *softirq*. N es el número máximo de paquetes en el sistema. Si hay N paquetes en el sistema, los nuevos paquetes entrantes serán rechazados.

Con objeto de obtener los parámetros de rendimiento con dichas suposiciones, en primer lugar, se planteará la cadena de Markov embebida para obtener su distribución en régimen estacionario; en segundo lugar, se calculará la distribución de la longitud de la cola del proceso en tiempo continuo; finalmente, se extraerán los parámetros de rendimiento más significativos.

4.5.1. Cadena de Markov embebida

Se examina el sistema en instantes de tiempo $\{t_0, t_1, \dots\}$ que son puntos de finalización de la rutina de servicio de *hardirq*, puntos de terminación de *vacation* o puntos donde se completa el servicio de cada paquete dentro de una *softirq*. Se define el espacio de estados del sistema $\mathcal{S} := \{L_i, \delta_i\}$ donde L_i es el número de paquetes del sistema en el punto embebido t_i. Si el punto t_i corresponde a un instante en el que finaliza una *hardirq*, $\delta_i = H$; si el punto embebido es un instante de terminación de *vacation*, $\delta_i = V$; finalmente, $\delta_i = m$ con $m = 1, 2, \dots, B$ indica que el punto t_i es el instante en el que se completa el servicio al paquete número m en la *softirq* actual. Las transiciones de estado en el proceso $\{L_i, \delta_i\}$ ocurren en

instantes de finalización de *hardirq*, instantes de terminación de *vacation* y en instantes de salida de paquetes del sistema.

Obsérvese que podría haber una ambigüedad en el punto embebido con $(L_i = 1, \delta_i = V)$. Podría ser el punto de terminación de la *vacation* con búfer vacío; es decir, cuando llega el primer paquete y esto causa el final de la *vacation* con búfer vacío y el comienzo de una *hardirq*. Sin embargo, ese punto embebido también podría hacer referencia a la finalización de un periodo de *vacation* "normal" con un paquete en el búfer; en este caso, el sistema comenzará una *softirq* después de esa *vacation* "normal". Por esa razón, se define un punto embebido con $(L_i = 1, \delta_i = V_0)$ que corresponde a la terminación de *vacation* con búfer vacío seguido de una *hardirq*, mientras que el conjunto de puntos embebidos $(L_i = n, \delta_i = V)$ con $n = 1, 2, \ldots, N$ está relacionado con puntos de terminación de *vacation* que son seguidas por una *softirq*.

Cuando el sistema está en régimen estacionario, las distribuciones de probabilidades $p_{n,m}$ se definen de la siguiente forma:

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

Teniendo en cuenta que los paquetes llegan al sistema según un proceso de Poisson, también se definen las probabilidades u_i, v_i y w_i.

u_i es la probabilidad de que i paquetes lleguen durante un tiempo H de *hardirq*.

$$u_i = \int_0^\infty e^{-\lambda x} \frac{(\lambda x)^i}{i!} dH(x) \tag{4.37}$$

v_i es la probabilidad de que i paquetes lleguen durante un tiempo V de *vacation*.

$$v_i = \int_0^\infty e^{-\lambda x} \frac{(\lambda x)^i}{i!} dV(x) \tag{4.38}$$

w_i es la probabilidad de que i paquetes lleguen durante un tiempo S de procesado de paquete en *softirq*.

$$w_i = \int_0^\infty e^{-\lambda x} \frac{(\lambda x)^i}{i!} dS(x) \tag{4.39}$$

También se definen las probabilidades u_k^c, v_k^c y w_k^c.

$$u_k^c = \sum_{i=k}^{\infty} u_i, \qquad v_k^c = \sum_{i=k}^{\infty} v_i, \qquad w_k^c = \sum_{i=k}^{\infty} w_i \tag{4.40}$$

Si denotamos con $H^{*(i)}$, $V^{*(i)}$ y $S^{*(i)}$ la derivada i-ésima de H^*, V^* y S^*, entonces

$$u_i = \left[\frac{(-\lambda)^i}{i!}\right] H^{*(i)}(\lambda), \quad v_i = \left[\frac{(-\lambda)^i}{i!}\right] V^{*(i)}(\lambda), \quad w_i = \left[\frac{(-\lambda)^i}{i!}\right] S^{*(i)}(\lambda) \tag{4.41}$$

donde $H^*(\theta)$, $V^*(\theta)$ y $S^*(\theta)$ son las transformadas de Laplace-Stieltjes de $H(x)$, $V(x)$ y $S(x)$ respectivamente.

Todas estas probabilidades satisfacen un conjunto de ecuaciones relacionado con la cadena de Markov embebida. A continuación, se enuncian dichas ecuaciones, separándolas en diferentes subconjuntos.

En puntos de terminación de la rutina de servicio de *hardirq*:

$$p_{n,H} = p_{1,V0} u_{n-1} \quad 0 \le n \le N-1 \tag{4.42a}$$

$$p_{N,H} = p_{1,V0} u^C_{N-1} \tag{4.42b}$$

En puntos de terminación de *vacation* que son seguidas por una *softirq*:

$$p_{n,V} = \sum_{k=1}^{n} v_{n-k} \left(p_{k,H} + p_{k,B}\right) \quad 1 \le n \le N \tag{4.43a}$$

$$p_{N,V} = \sum_{k=1}^{N-1} v^C_{N-k} \left(p_{k,H} + p_{k,B}\right) + p_{N,H} \tag{4.43b}$$

En puntos donde se completa el servicio a un paquete dentro de una *softirq*:

$$p_{n,1} = \sum_{k=1}^{n+1} w_{n-k+1} p_{k,V}, \quad 0 \le n \le N-2 \tag{4.44a}$$

$$p_{N-1,1} = \sum_{k=1}^{N} w^C_{N-k} p_{k,V} \tag{4.44b}$$

$$p_{n,m} = \sum_{k=1}^{n+1} w_{n-k+1} p_{k,m-1}, \quad 0 \le n \le N-2; \;\; 2 \le m \le B \tag{4.44c}$$

$$p_{N-1,m} = \sum_{k=1}^{N-1} w^C_{N-k} p_{k,m-1}, \quad 2 \le m \le B \tag{4.44d}$$

Finalmente, en el punto de terminación de *vacation* con búfer vacío:

$$p_{1,V0} = \sum_{m=1}^{B} p_{0,m} \tag{4.45}$$

También se tiene la ecuación de la suma de probabilidades

$$\sum_{n=1}^{N} p_{n,H} + \sum_{n=1}^{N} p_{n,V} + \sum_{n=0}^{N-1} \sum_{m=1}^{B} p_{n,m} + p_{1,V0} = 1 \tag{4.46}$$

El paso siguiente consiste en la resolución de la probabilidades $p_{n,m}$ en régimen estacionario de la cadena de Markov embebida. Para ello, se reescriben las Ecuaciones (4.42-4.46) en forma matricial como función de los coeficientes u_i, v_i, w_i, u_i^c, v_i^c y w_i^c. Aplicando un método análogo al usado para calcular las probabilidades de estado del modelo M2, se obtienen las probabilidades $p_{n,m}$.

Una vez que se tienen las probabilidades $p_{n,m}$, se pueden establecer unos parámetros de sistema que serán utilizados posteriormente para estimar parámetros de rendimiento. En primer lugar, se define b_0 como la probabilidad de que un punto embebido sea un punto de terminación de *vacation* previo a una *hardirq*; en segundo lugar, b_1 como la probabilidad de que un punto embebido sea un punto de terminación de *hardirq*; en tercer lugar, b_2 como la probabilidad de que un punto embebido sea un punto de terminación de *vacation* previo a una *softirq*; y, finalmente, b_3 como la probabilidad de que un punto embebido sea un punto de en el que se completa el servicio de un paquete dentro de una *softirq*. Por tanto, se tiene:

$$b_0 = p_{1,V0} \; ; \quad b_1 = \sum_{n=1}^{N} p_{n,H} \; ; \quad b_2 = \sum_{n=1}^{N} p_{n,V} \; ; \quad b_3 = \sum_{n=0}^{N-1} \sum_{m=1}^{m=B} p_{n,m} \tag{4.47}$$

Otro parámetro de interés es la frecuencia de los puntos embebidos que se denota como σ. Se interpreta la inversa de σ como la media del intervalo entre dos puntos embebidos consecutivos. σ puede expresarse como:

$$\sigma^{-1} = b_0 \frac{1}{\lambda} + b_1 E(H) + b_2 E(V) + b_3 E(S) \tag{4.48}$$

4.5.2. Distribución general de la longitud de la cola

A continuación, se desarrolla el cálculo de la distribución de probabilidades de la cola que puede considerarse de tipo M/G/1/N con rutinas de servicio de *hardirq*, *vacations* y procesado de paquetes en *softirq* con disciplina de servicio limitada por *budget* B.

Sea la variable L el número de paquetes del sistema en régimen estación en un instante de tiempo arbitrario, siendo $(0 \leq L \leq N)$. Sea la variable δ el estado del sistema desde el punto de vista del procesador. Así, si $\delta = H$, el procesador está tendiendo a una rutina de servicio de *hardirq*; si $\delta = V$, el procesador está en *vacation* con $L > 0$; si $\delta = V_0$, el procesador también está en *vacation*, pero la cola está vacía $(L = 0)$ y esta *vacation* terminará cuando el primer paquete llegue al sistema; finalmente, si $\delta = m$ siendo $1 \leq m \leq B$, esto indica que el procesador está sirviendo al paquete número m en la *softirq* actual.

Para obtener la distribución general de la longitud de la cola del proceso continuo se utiliza la técnica de la variable suplementaria [147]. Por ello, se consideran las siguientes variables suplementarias:

- $\hat{H}$ = tiempo de servicio de *hardirq* de recurrencia hacia adelante, es decir, el tiempo de servicio restante para la *hardirq*.
- $\hat{V}$ = tiempo de *vacation* de recurrencia hacia adelante, es decir, el tiempo que resta para que termine la *vacation* actual, no estando el búfer vacío ($n > 0$).
- $\hat{V}_0$ = tiempo de *vacation* de recurrencia hacia adelante cuando no hay ningún paquete en el búfer ($n = 0$), es decir, el tiempo restante de *vacation* que finalizará cuando llegue el primer paquete; debido a que la llegada de paquetes sigue un proceso de Poisson, $\hat{V}_0$ tiene una distribución exponencial de media $1/\lambda$.
- $\hat{S}$ = tiempo de servicio de paquete (en *softirq*) de recurrencia hacia adelante, es decir, el tiempo de servicio restante para la salida de un paquete dentro de la ejecución de la *softirq* actual.

Asimismo, se definen los siguientes tiempos de servicio de recurrencia hacia atrás:

- $\tilde{H}$ = tiempo de servicio de recurrencia hacia atrás en *hardirq*, es decir, el tiempo de servicio transcurrido en *hardirq*.
- $\tilde{V}$ = tiempo de recurrencia hacia atrás en *vacation* ($n > 0$), es decir, el tiempo de *vacation* transcurrido para el procesador, no estando el búfer vacío ($n > 0$).
- $\tilde{V}_0$ = tiempo de recurrencia hacia atrás en *vacation* con ($n = 0$), es decir, el tiempo de *vacation* transcurrido para el procesador, estando el búfer vacío ($n = 0$).
- $\tilde{S}$ = tiempo de servicio de recurrencia hacia atrás para un paquete en *softirq*, es decir, el tiempo de servicio transcurrido para el paquete que está siendo servicio en la *softirq*.

Se pretende determinar las expresiones de las probabilidades $\pi_{n,H}$, $\pi_{n,V}$ y $\pi_{n,m}$, partiendo de

$$\pi_{n,H}(\tau)d\tau = Pr\left\{L = n, \delta = H, \tau < \hat{H} \leq \tau + d\tau\right\}, \quad 1 \leq n \leq N \tag{4.49a}$$

$$\pi_{0,V}(\tau)d\tau = Pr\left\{L = 0, \delta = V_0, \tau < \hat{V}_0 \leq \tau + d\tau\right\} \tag{4.49b}$$

$$\pi_{n,V}(\tau)d\tau = Pr\left\{L = n, \delta = V, \tau < \hat{V} \leq \tau + d\tau\right\}, \quad 1 \leq n \leq N \tag{4.49c}$$

$$\pi_{n,m}(\tau)d\tau = Pr\left\{L = n, \delta = m, \tau < \hat{S} \leq \tau + d\tau\right\}, \quad 1 \leq n \leq N;\ 1 \leq m \leq B \tag{4.49d}$$

y sabiendo que sus correspondiente transformadas de Laplace-Stieltjes son:

$$\pi^*_{n,H}(\theta) = \int_0^\infty e^{-\theta\tau}\pi_{n,H}(\tau)d\tau, \qquad 1 \le n \le N \tag{4.50a}$$

$$\pi^*_{n,V}(\theta) = \int_0^\infty e^{-\theta\tau}\pi_{n,V}(\tau)d\tau, \qquad 0 \le n \le N \tag{4.50b}$$

$$\pi^*_{n,m}(\theta) = \int_0^\infty e^{-\theta\tau}\pi_{n,m}(\tau)d\tau, \qquad 1 \le n \le N;\ 1 \le m \le B \tag{4.50c}$$

Las integrales de las transformadas de Laplace-Stieltjes (4.50a-4.50c) pueden ser evaluadas condicionándolas en $\chi(\tilde{H})$, $\chi(\tilde{V})$ y $\chi(\tilde{S})$, el número de paquetes que llegan durante $\tilde{H}$, el tiempo de servicio transcurrido en *hardirq*, durante $\tilde{V}$, el tiempo transcurrido en *vacation*, y durante $\tilde{S}$, el tiempo de servicio transcurrido con un paquete en *softirq*. Por ello, se tienen los siguientes casos:

- Probabilidades relacionadas con la rutina de servicio de *hardirq*:

$$\pi^*_{n,H}(\theta) = Pr\{\delta = H\} \cdot E\left[e^{-\theta\hat{H}}|\chi(\tilde{H} = n-1)\right] \cdot Pr\left\{\chi(\tilde{H}) = n-1\right\} \qquad 1 \le n \le N-1 \tag{4.51a}$$

$$\pi^*_{N,H}(\theta) = Pr\{\delta = H\} \sum_{k=N-1}^{\infty} E\left[e^{-\theta\hat{H}}|\chi(\tilde{H} = k)\right] Pr\left\{\chi(\tilde{H}) = k)\right\} \tag{4.51b}$$

- Probabilidades relacionadas con *vacations*:

$$\pi^*_{0,V}(\theta) = Pr\{\delta = V_0\} \frac{\sum_{m=1}^{B} p_{0,m} E\left[e^{-\theta\tilde{V_0}}|\chi(\tilde{V_0} = 0)\right] Pr\left\{\chi(\tilde{V_0}) = 0\right\}}{\sum_{m=1}^{B} p_{0,m}} \tag{4.52a}$$

$$\pi^*_{n,V}(\theta) = Pr\{\delta = V\} \cdot \frac{\sum_{j=1}^{n}(p_{j,H} + p_{j,B}) E\left[e^{-\theta\hat{V}}|\chi(\tilde{V} = n-j)\right] Pr\left\{\chi(\tilde{V}) = n-j\right\}}{\sum_{j=1}^{N} p_{j,H} + \sum_{j=1}^{N-1} p_{j,B}} \qquad 1 \le n \le N \tag{4.52b}$$

$$\pi^*_{N,V}(\theta) = Pr\{\delta = V\} \left(\frac{\sum_{j=1}^{N-1}(p_{j,H} + p_{j,B}) \sum_{k=N-j}^{\infty} E\left[e^{-\theta\hat{V}}|\chi(\tilde{V} = k)\right] Pr\left\{\chi(\tilde{V}) = k\right\}}{\sum_{j=1}^{N} p_{j,H} + \sum_{j=1}^{N-1} p_{j,B}} + \right.$$
$$\left. + \frac{p_{N,H} \sum_{k=0}^{\infty} E\left[e^{-\theta\hat{V}}|\chi(\tilde{V} = k)\right] Pr\left\{\chi(\tilde{V}) = k\right\}}{\sum_{j=1}^{N} p_{j,H} + \sum_{j=1}^{N-1} p_{j,B}} \right) \tag{4.52c}$$

- Probabilidades asociadas al tratamiento de paquetes en *softirq*:

$$\pi^*_{n,1}(\theta) = Pr\{\delta = 1\} \frac{\sum_{j=1}^{n} p_{j,V} E\left[e^{-\theta \hat{S}} | \chi(\tilde{S} = n-j)\right] Pr\left\{\chi(\tilde{S}) = n-j\right\}}{\sum_{j=1}^{N} p_{j,V}}$$

$$1 \leq n \leq N-1 \quad (4.53a)$$

$$\pi^*_{N,1}(\theta) = Pr\{\delta = 1\} \frac{\sum_{j=1}^{N} p_{j,V} \sum_{k=N-j}^{\infty} E\left[e^{-\theta \hat{S}} | \chi(\tilde{S} = k)\right] Pr\left\{\chi(\tilde{S}) = k\right\}}{\sum_{j=1}^{N} p_{j,V}} \quad (4.53b)$$

$$\pi^*_{n,m}(\theta) = Pr\{\delta = m\} \frac{\sum_{j=1}^{n} p_{j,m-1} E\left[e^{-\theta \hat{S}} | \chi(\tilde{S} = n-j)\right] Pr\left\{\chi(\tilde{S}) = n-j\right\}}{\sum_{j=1}^{N-1} p_{j,m-1}}$$

$$1 \leq n \leq N-1;\ 2 \leq m \leq B \quad (4.53c)$$

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

$$2 \leq m \leq B \quad (4.53d)$$

Las probabilidades $Pr\{\delta\}$ pueden expresarse como función de las probabilidades de la cadena de Markov embebida, de los tiempos medios de *hardirq*, de *vacation* y de servicio de paquetes en *softirq*, así como del parámetro σ (frecuencia de los puntos embebidos).

$$Pr\{\delta = V_0\} = \frac{\frac{1}{\lambda} p_{1,V0}}{\sigma^{-1}} = \frac{\sigma}{\lambda} p_{1,V0} = \frac{\sigma}{\lambda} \sum_{m=1}^{B} p_{0,m} \quad (4.54a)$$

$$Pr\{\delta = H\} = \frac{E[H] \sum_{n=1}^{N} p_{n,H}}{\sigma^{-1}} = \sigma E[H] \sum_{n=1}^{N} p_{n,H} \quad (4.54b)$$

$$Pr\{\delta = V\} = \frac{E[V] \sum_{n=1}^{N} p_{n,V}}{\sigma^{-1}} = \sigma E[V] \sum_{n=1}^{N} p_{n,V} \quad (4.54c)$$

$$Pr\{\delta = m\} = \frac{E[S] \sum_{n=0}^{N-1} p_{n,m}}{\sigma^{-1}} = \sigma E[S] \sum_{n=0}^{N-1} p_{n,m} \quad (4.54d)$$

Después de realizar el correspondiente desarrollo algebraico, se llega a las siguientes expresiones:

$$\pi^*_{0,V}(\theta) = \frac{\sigma}{\lambda} p_{1,V0} \left(\frac{\lambda}{\lambda+\theta}\right)\left(\frac{\lambda}{\lambda-\theta}\right) \tag{4.55a}$$

$$\pi^*_{n,H}(\theta) = \frac{\sigma}{\lambda}\left\{H^*(\theta)\left(\sum_{j=1}^{N} p_{j,H}\right)\left(\frac{\lambda}{\lambda-\theta}\right)^n - \sum_{j=1}^{n} p_{j,H}\left(\frac{\lambda}{\lambda-\theta}\right)^{n-j+1}\right\}$$
$$1 \le n \le N-1 \tag{4.55b}$$

$$\pi^*_{N,H}(\theta) = \frac{-\sigma}{\theta}\left\{H^*(\theta)\left(\sum_{j=1}^{N} p_{j,H}\right)\left(\frac{\lambda}{\lambda-\theta}\right)^{N-1} - \sum_{j=1}^{N} p_{j,H}\left(\frac{\lambda}{\lambda-\theta}\right)^{N-j}\right\} \tag{4.55c}$$

$$\pi^*_{n,V}(\theta) = \frac{\sigma}{\lambda}\left\{V^*(\theta)\sum_{j=1}^{n}(p_{j,H}+p_{j,B})\left(\frac{\lambda}{\lambda-\theta}\right)^{n-j+1} - \sum_{j=1}^{n} p_{j,V}\left(\frac{\lambda}{\lambda-\theta}\right)^{n-j+1}\right\}$$
$$1 \le n \le N-1 \tag{4.55d}$$

$$\pi^*_{N,V}(\theta) = \frac{-\sigma}{\theta}\left\{V^*(\theta)\sum_{j=1}^{N} p_{j,H}\left(\frac{\lambda}{\lambda-\theta}\right)^{N-j} + V^*(\theta)\sum_{j=1}^{N-1} p_{j,B}\left(\frac{\lambda}{\lambda-\theta}\right)^{N-j} - \right.$$
$$\left. - \sum_{j=1}^{N} p_{j,V}\left(\frac{\lambda}{\lambda-\theta}\right)^{N-j}\right\} \tag{4.55e}$$

$$\pi^*_{n,1}(\theta) = \frac{\sigma}{\lambda}\left\{S^*(\theta)\sum_{j=1}^{n} p_{j,V}\left(\frac{\lambda}{\lambda-\theta}\right)^{n-j+1} - \sum_{j=0}^{n-1} p_{j,1}\left(\frac{\lambda}{\lambda-\theta}\right)^{n-j}\right\}$$
$$1 \le n \le N-1 \tag{4.55f}$$

$$\pi^*_{N,1}(\theta) = \frac{-\sigma}{\theta}\left\{S^*(\theta)\sum_{j=1}^{N} p_{j,V}\left(\frac{\lambda}{\lambda-\theta}\right)^{N-j} - \sum_{j=0}^{N-1} p_{j,1}\left(\frac{\lambda}{\lambda-\theta}\right)^{N-j-1}\right\} \tag{4.55g}$$

$$\pi^*_{n,m}(\theta) = \frac{\sigma}{\lambda}\left\{S^*(\theta)\sum_{j=1}^{n} p_{j,m-1}\left(\frac{\lambda}{\lambda-\theta}\right)^{n-j+1} - \sum_{j=0}^{n-1} p_{j,m}\left(\frac{\lambda}{\lambda-\theta}\right)^{n-j}\right\}$$
$$1 \le n \le N-1;\ 2 \le m \le B \tag{4.55h}$$

$$\pi^*_{N,m}(\theta) = \frac{-\sigma}{\theta}\left\{S^*(\theta)\sum_{j=1}^{N-1} p_{j,m-1}\left(\frac{\lambda}{\lambda-\theta}\right)^{N-j} - \sum_{j=0}^{N-1} p_{j,m}\left(\frac{\lambda}{\lambda-\theta}\right)^{N-j-1}\right\}$$
$$2 \le m \le B \tag{4.55i}$$

Dado que σ^{-1} es el intervalo medio entre dos puntos embebidos consecutivos, se interpreta la expresión σ/λ, que aparece en las ecuaciones anteriores, como la inversa del número medio de llegadas entre puntos embebidos.

Finalmente, para obtener la distribución general de la longitud de la cola, se introduce $\theta = 0$ en las expresiones de $\pi^*_{n,m}(\theta)$, donde $n \in \{0, 1, \ldots, N\}$ y $m \in \{H, V, 1, 2, \ldots, B\}$, de las Ecuaciones (4.55a-4.55i), esto es, $\pi_{n,m} = \pi^*_{n,m}(\theta = 0)$.

$$\pi_{0,V} = \frac{\sigma}{\lambda} p_{1,V0} \tag{4.56a}$$

$$\pi_{n,H} = \frac{\sigma}{\lambda} \left\{ p_{1,V0} - \sum_{j=1}^{n} p_{j,H} \right\}, \quad 1 \leq n \leq N-1 \tag{4.56b}$$

$$\pi_{N,H} = \frac{\sigma}{\lambda} \left\{ \lambda E(H) p_{1,V0} - (N-1) p_{1,V0} + \sum_{j=1}^{N} (N-j) p_{j,H} \right\} \tag{4.56c}$$

$$\pi_{n,V} = \frac{\sigma}{\lambda} \left\{ \sum_{j=1}^{n} (p_{j,H} + p_{j,B}) - \sum_{j=1}^{n} p_{j,V} \right\}, \quad 1 \leq n \leq N-1 \tag{4.56d}$$

$$\pi_{N,V} = \frac{\sigma}{\lambda} \left\{ \lambda E(V) \sum_{j=1}^{N} (p_{j,H} + p_{j,B}) - \sum_{j=1}^{N} (p_{j,H} + p_{j,B}) (N-j) + \right.$$
$$\left. + \sum_{j=1}^{N} p_{j,V} (N-j) \right\} \tag{4.56e}$$

$$\pi_{n,1} = \frac{\sigma}{\lambda} \left\{ \sum_{j=1}^{n} p_{j,V} - \sum_{j=0}^{n-1} p_{j,1} \right\}, \quad 1 \leq n \leq N-1 \tag{4.56f}$$

$$\pi_{N,1} = \frac{\sigma}{\lambda} \left\{ \lambda E(S) \sum_{j=1}^{N} p_{j,V} - \sum_{j=1}^{N} p_{j,V} (N-j) + \sum_{j=0}^{N-1} p_{j,1} (N-j-1) \right\} \tag{4.56g}$$

$$\pi_{n,m} = \frac{\sigma}{\lambda} \left\{ \sum_{j=1}^{n} p_{j,m-1} - \sum_{j=0}^{n-1} p_{j,m} \right\}, \quad 1 \leq n \leq N-1; \;\; 2 \leq m \leq B \tag{4.56h}$$

$$\pi_{N,m} = \frac{\sigma}{\lambda} \left\{ \lambda E(S) \sum_{j=1}^{N-1} p_{j,m-1} - \sum_{j=1}^{N-1} (N-j) p_{j,m-1} + \right.$$
$$\left. + \sum_{j=0}^{N-1} (N-j-1) p_{j,m} \right\} \quad 2 \leq m \leq B \tag{4.56i}$$

4.5.3. Parámetros de rendimiento

Después de calcular las probabilidades de la cadena de Markov embebida y la distribución conjunta de la longitud de las colas, se pueden determinar los parámetros de rendimiento del modelo M3 como probabilidad de bloqueo, throughput de captura, frecuencia de *softirq*...

4.5.3.1. Probabilidad de bloqueo

La probabilidad de bloqueo (P_B) del modelo M3 puede ser calculada con la misma fórmula utilizada en el modelo M2, esto es, sumando las probabilidades de los estados de bloqueo (N, m) con $m \in \{H, V, 1, 2, \ldots, B\}$.

$$P_B = \pi_{N,H} + \pi_{N,V} + \sum_{m=1}^{B} \pi_{N,m} \tag{4.57}$$

También se puede obtener la probabilidad de bloqueo con los resultados derivados del análisis de la cadena de Markov embebida. Si se considera la *softirq* como la principal actividad del modelo, ya que la salida de paquetes del sistema es el resultado del procesado de *softirq*. Se puede definir $\rho = \lambda E(S)$ como la carga ofrecida al sistema y $\rho\prime$ como carga transportada, esto es, la fracción del tiempo en la que la *softirq* está activa.

$$\rho\prime = \frac{b_3 E(S)}{b_1 E(H) + b_2 E(V) + b_3 E(S) + b_0 \frac{1}{\lambda}} = \sigma b_3 E(S) \tag{4.58}$$

La probabilidad de bloqueo también viene dada como:

$$P_B = \frac{(\rho - \rho\prime)}{\rho} = 1 - \frac{\rho\prime}{\rho} = 1 - \frac{\sigma b_3}{\lambda} = 1 - \frac{\sigma}{\lambda} \sum_{n=0}^{N-1} \sum_{m=1}^{B} p_{n,m} \tag{4.59}$$

Se ha comprobado que las dos expresiones de la probabilidad de bloqueo P_B son equivalentes. Si se introducen las expresiones (4.56c), (4.56e), (4.56i) de $\pi_{N,H}$, $\pi_{N,V}$ y $\pi_{N,m}$ respectivamente en la expresión (4.57) de P_B, el resultado es el mismo que el de la expresión (4.59) de P_B.

4.5.3.2. Throughput de captura

El throughput de captura (X_C) del modelo M3 puede calcularse a partir de la tasa λ y la probabilidad de bloqueo P_B, obtenida a través de la expresión (4.57) o de la (4.59).

$$X_C = \lambda(1 - P_B) \tag{4.60}$$

4.5.3.3. Utilización de CPU

Se estima el parámetro U_C, utilización de CPU, con las probabilidades de *hardirq* y de *softirq* en régimen estacionario, aplicando la misma expresión que la del modelo M2.

$$U_C = \sum_{n=1}^{N} \pi_{n,H} + \sum_{n=1}^{N} \sum_{m=1}^{B} \pi_{n,m} \tag{4.61}$$

4.5.3.4. Frecuencia de hardirq

Al igual que se hizo para el modelo M2, aplicando los mismos conceptos de tiempo medio de ciclo T_{cycle}, tiempo medio de *softirq* $T_{softirq}$ y los tiempos exponenciales entre llegadas debido a su carácter poissoniano, la frecuencia de *hardirq*, $f_{hardirq}$, viene dada por:

$$f_{hardirq} = \frac{1}{T_{cycle}} = \lambda \cdot \pi_{0,V} \tag{4.62}$$

4.5.3.5. Frecuencia de softirq

Con el mismo razonamiento utilizado para el modelo M2, esta es la expresión de la frecuencia de *softirq*, $f_{softirq}$, para el modelo M3.

$$f_{softirq} = \frac{\sum_{n=1}^{N} \pi_{n,V}}{E(V)} \tag{4.63}$$

4.5.3.6. Tiempo medio de softirq

El tiempo medio de *softirq*, $T_{softirq}$, del modelo M3 se obtiene con:

$$T_{softirq} = \frac{T_{cycle} \sum_{n=1}^{N} \sum_{m=1}^{B} \pi_{n,m}}{k_s} = \frac{E(V) \sum_{n=1}^{N} \sum_{m=1}^{B} \pi_{n,m}}{\sum_{n=1}^{N} \pi_{n,V}} \tag{4.64}$$

4.5.3.7. Número medio de paquetes de una softirq

Finalmente, se calcula el número medio de paquetes procesados dentro de una *softirq*, $\overline{m}_{softirq}$, de la siguiente forma:

$$\overline{m}_{softirq} = \frac{T_{softirq}}{E(S)} = \frac{E(V) \sum_{n=1}^{N} \sum_{m=1}^{B} \pi_{n,m}}{E(S) \sum_{n=1}^{N} \pi_{n,V}} \tag{4.65}$$

También es posible estimar $\overline{m}_{softirq}$ usando las probabilidades de la cadena de Markov embebida.

$$\overline{m}_{softirq} = \frac{\sum_{m=1}^{B} m p_{0,m} + B \sum_{n=1}^{N-1} p_{n,B}}{\sum_{m=1}^{B} p_{0,m} + \sum_{n=1}^{N-1} p_{n,B}} \tag{4.66}$$

4.6. Evaluación de los modelos con vacations

Este apartado presenta resultados de los modelos M1, M2 y M3. Para llegar a ellos, se han implementado en MATLAB las ecuaciones derivadas de los modelos y, tras introducir ciertos valores de entrada requeridos por los modelos y ejecutar el programa de cálculo, se ha obtenido un conjunto de resultados analíticos que se representan gráficamente. Asimismo, con objeto de realizar comparativas, al igual que se hizo en la validación del capítulo anterior, se presentan valores experimentales obtenidos con la sonda real de captura y análisis de paquetes basada en Linux [40] que opera dentro de la plataforma de medidas de laboratorio [130]. Para que sean comparables ambos tipos de resultados, algunos parámetros de entrada de los modelos (λ, $E(S)$, $E(V)$, $E(H)$) toman sus valores a partir de mediciones realizadas sobre la plataforma experimental.

4.6.1. Escenarios de evaluación

Para llevar a cabo la evaluación de los modelos, se definen tres escenarios de evaluación: V1, V2 y V3. Cada uno de ellos se caracteriza por tener distinto comportamiento de *vacation*. Esto quiere decir que los tiempos de *vacation*, representados por la variable V en el modelo, serán distintos en cada escenario.

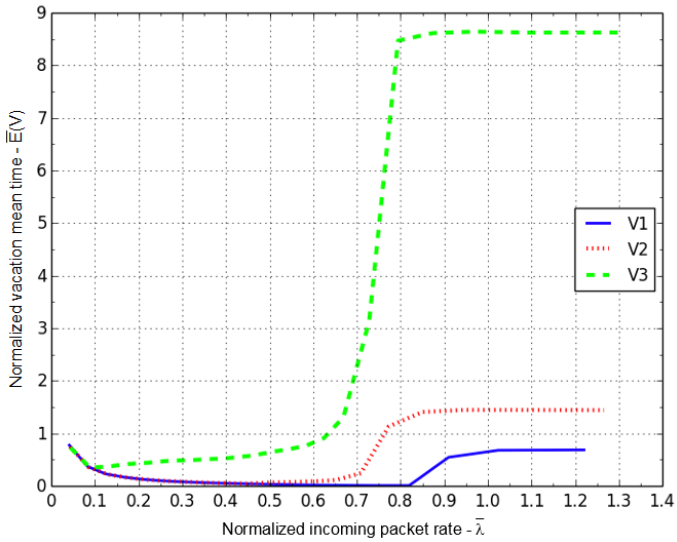


FIGURA 4.7: Escenarios de evaluación V1, V2 y V3.

Al igual que ocurría en la validación del modelo del Capítulo 3, algunos de los parámetros de entrada se establecen basándose en mediciones sobre un equipo real. En este caso, sobre la misma plataforma de pruebas del modelo del Capítulo 3 (ver Figura3.4) y con la misma sonda de análisis de tráfico de red, se toman medidas experimentales de:

- Tasa de paquetes entrantes al sistema de captura.
- Tiempos de procesado de *hardirq*.
- Tiempos de procesado de *softirq*.
- Se considera tiempo de *vacation* a todo aquel en el que el procesador no está tratando ni *hardirqs* ni *softirqs*.

La Figura 4.7 muestra la representación gráfica asociada a los tiempos de *vacation* de los tres escenarios. 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*, $\overline{E}(V) = (1/\mu_V)\,/\,(B/\mu_S)$, con $B = 300$ ya que este es el valor típico de *budget*. En la Figura 4.7 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 esta, las operaciones de análisis posteriores a la captura requieren mayores consumos computacionales a medida que aumenta la tasa de entrada de paquetes. Así, V1 corresponde al escenario de carga de análisis baja, V2 al escenario de carga de análisis intermedia y V3 al escenario de carga de análisis alta. Cuanto mayor es la carga de análisis, mayor es el tiempo requerido por la tarea que el procesador debe realizar en el tiempo de vacation.

4.6.2. Resultados de evaluación

Como se ha mencionado previamente, se tienen resultados de rendimiento de los modelos analíticos M1, M2, M3, así como medidas experimentales provenientes de la sonda real de laboratorio que captura paquetes. los valores experimentales aparecerán referidas como "Lab" en las gráficas de este apartado. A continuación, se muestran algunos ejemplos de resultados.

En primer lugar, la Figura 4.8 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 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

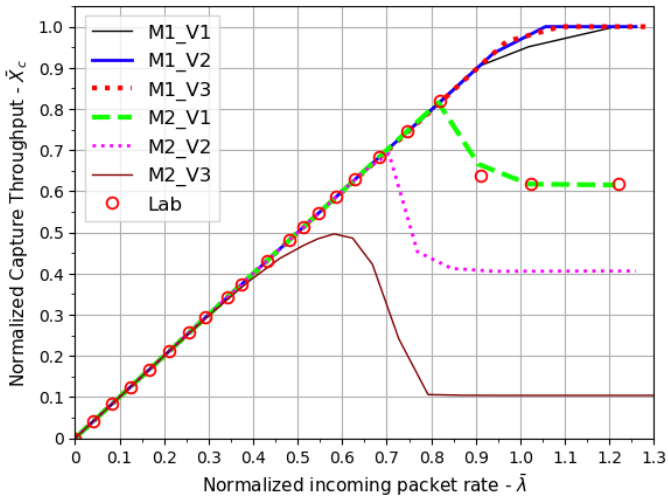


FIGURA 4.8: Resultados de throughput de captura normalizado para diversos escenarios (V1, V2, y V3).

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.

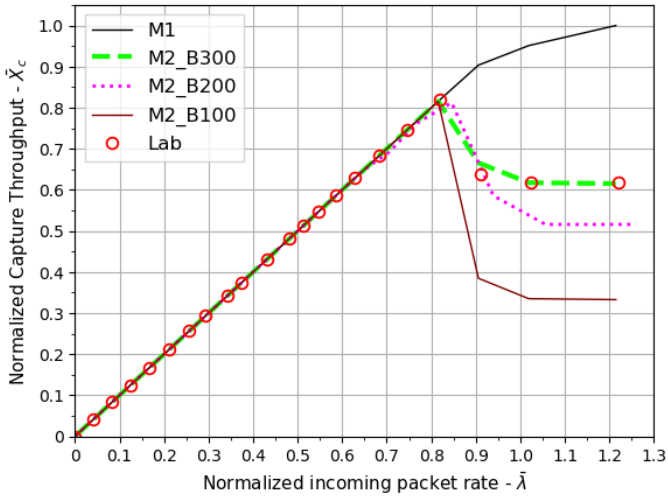


FIGURA 4.9: Resultados de throughput de captura normalizado para difentes budgets y escenario V1.

En segundo lugar, la Figura 4.9 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.

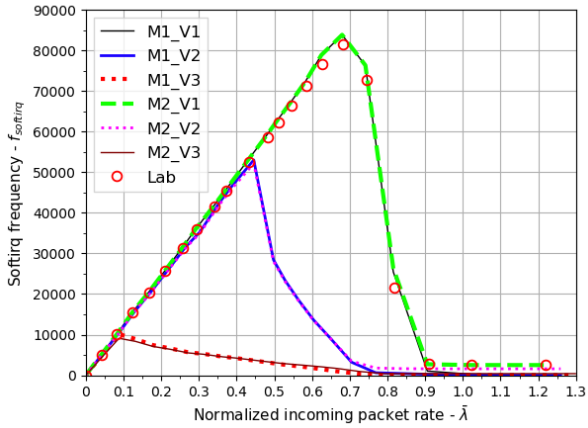


FIGURA 4.10: Resultados de frecuencia de softirq para diversos escenarios (V1, V2 y V3).

A continuación, la Figura 4.10 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 los 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 Figura 4.10 también se da que el caso del modelo M2 con el escenario 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 $f_{hirq} \to 0$.

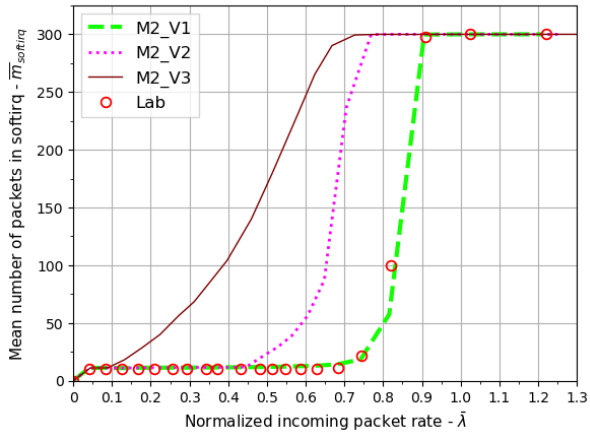


FIGURA 4.11: Resultados de número medio de paquetes por softirq para diversos escenarios (V1, V2 y V3).

La Figura 4.11 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.

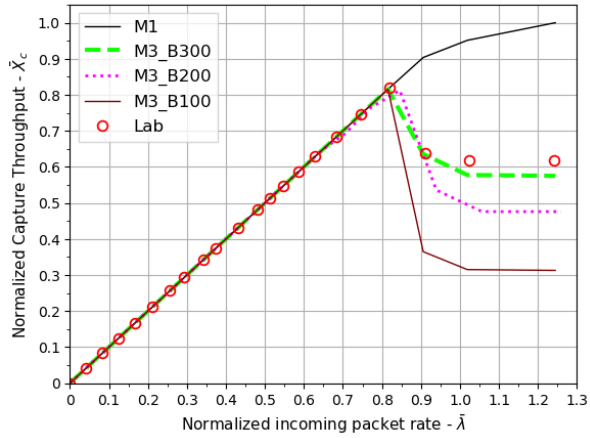


FIGURA 4.12: Throughput de análisis del modelo M3 en escenario V1.

Por último, se muestran dos gráficas relacionadas con el modelo M3. La Figura 4.12 muestra el throughput del modelo M3 para distintos valores de *budget*, B, sobre el escenario V1. Corresponde a distribuciones de tiempo deterministas para el procesamiento de *hardirq*, para el tratamiento de

paquete en *softirq* y para el tiempo de *vacation*. Si se compara con el resultado obtenido para el modelo M2 (Figura 4.9), se observa que los valores de throughput no son iguales, pero sí muy parecidos. Por tanto, el comportamiento de ambos casos es similar.

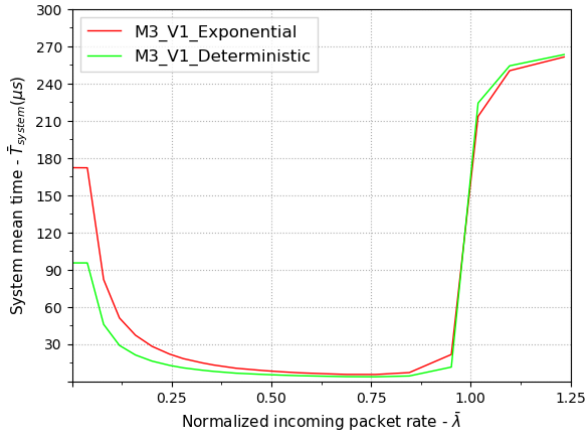


FIGURA 4.13: Retardo medio del sistema del modelo M3 con escenario V1.

La Figura 4.13 muestra el retardo medio que experimenta un paquete en el sistema, desde que entra al búfer hasta que sale del proceso de la *softirq*. Este retardo medio, $E(T)$, se calcula a partir de la expresión de Little, $E(T) = E(n)/X_C$ donde $E(n)$ es el número medio de paquetes en el sistema, que se puede calcular a partir de las probabilidades de estado $\pi_{n,m}$, y X_C es el throughput del sistema de captura. Los resultados de la gráfica son del modelo M3 con escenario V1 para el caso de distribuciones exponenciales y para el caso de distribuciones deterministas. Cuando se evalúa el modelo M3 con tiempos de *hardirq*, de tratamiento de paquete en *softirq* y de *vacation* exponenciales, los resultados son similares al del modelo M2. Sin embargo, a diferencia de lo que ocurría con el throughput, si variamos la distribución de tiempo en el modelo M3, la gráfica del retardo medio es diferente. En la Figura 4.13 se observa que la mayor diferencia se produce con tasas bajas, donde la respuesta del sistema con tiempos deterministas es mejor. A medida que las tasas de entrada aumentan, las dos gráficas se parecen más, tienen la misma tendencia, aunque los valores no son iguales.

Para finalizar este análisis de resultados, 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.

4.7. Conclusiones

En este capítulo se ha propuesto un modelo analítico basado en un sistema de colas con *vacations* para analizar el rendimiento del sistema de captura de paquetes de Linux. El concepto de *vacation* permite modelar el comportamiento del procesador responsable de capturar paquetes así como otras tareas adicionales durante el tiempo de vacation.

El primer modelo propuesto, M1, puede considerarse como un modelo simple que prioriza el proceso de captura con una disciplina de servicio exhaustiva ($B \rightarrow \infty$). Si el sistema no está saturado, está garantizada la ejecución de las tareas adicionales en las vacations; sin embargo, bajo condiciones de saturación, el proceso de captura monopoliza el tiempo del procesador y se alcanza un throughput de captura aceptable en detrimento de realizar tareas adicionales.

El segundo y el tercer modelo, M2 y M3, son más complejos, y su disciplina de servicio limitada permite tener presentes algunas particularidades del sistema de captura de paquete de Linux (procesamiento *hardirq*, ejecución de *softirq* y *budget*). La limitación de *budget* dentro de la *softirq* permite asegurar la ejecución de otras tareas, aunque ello cause un pérdida de throughput de captura. En estos casos, es interesante evaluar si compensa tener *vacations* (para poder hacer tareas adicionales), aunque eso conlleve una pérdida de throughput de captura.

En general, los resultados obtenidos son satisfactorios, sobre todo, aquellos donde se ha podido contrastar el resultado del throughput del modelo con el throughput medido sobre una sonda real. Otro aspecto a valorar positivamente es que, mediante los resultados del modelo, se puede apreciar el efecto que tienen los tiempos de *vacations* sobre el throughput de la tarea principal (la etapa de captura). Visto esto, cabe pensar que este concepto de *vacation* también pueda ser utilizado en otros modelos que toman como base los consumos computacionales de funciones software.

Capítulo 5

Modelo de colas para servicios de Misión Crítica sobre red B5G

Tras haber presentado, en los capítulos 3 y 4, modelos basados en teoría de colas aplicados a una sonda de análisis de tráfico, este capítulo aborda el objetivo de integrar el conocimiento adquirido en esa investigación al entorno de las redes de futuro *Beyond 5G (B5G)*. Tal y como se argumenta en el estado del arte del Capítulo 2, el modelado basado en teoría de colas puede ayudar en el despliegue eficiente de un servicio virtualizado de Misión Crítica (MC). La provisión del servicio requiere una serie de funciones *VNF (Virtual Network Functions)*, que pueden ubicarse de forma distribuida sobre los recursos disponibles en la infraestructura de las redes 5G y B5G. En este capítulo, se propone un modelo que determina cuál es la mejor forma de desplegar las VNF, teniendo en cuenta el conjunto de indicadores *KPI (Key Performance Indicator)* que el servicio debe cumplir. Al igual que en los modelos presentados en los capítulos 3 y 4, se tomará como base el consumo computacional de funciones software, para construir un modelo de teoría de colas.

Este modelo permite representar las relaciones funcionales que hay entre las diferentes VNF y considera las restricciones que la topología de red impone en función de cómo están dispuestos los elementos físicos de la infraestructura. En concreto, se va a estudiar cómo sería el despliegue óptimo de un servicio de Misión Crítica *MC Push-To-Talk (MCPTT)*, considerando, además del envío de audio, las extensiones de envío de datos (*MCDATA*) y vídeo (*MCVIDEO*). Estos tres servicios constituyen lo que se denomina MCX-PTT. En el modelado, se considera también el tráfico del plano de control; tanto la señalización IMS, como la específica del estándar MCX-PTT de *floor control*, necesaria esta última para el arbitrio de la comunicación en ese entorno.

El capítulo sigue la estructura que se especifica a continuación. En primer lugar, el apartado 5.1 expone la problemática de la distribución de

las funcionalidades de los servicios MCX e identifica los componentes que serán representados en el modelo. A continuación, en el apartado 5.2, se describe la propuesta de modelo de red de colas M/M/1, para luego, en el apartado 5.3 resolver el problema de optimización que determina cómo desplegar las funciones VNF apropiadamente. Para demostrar la utilidad del modelo, el apartado 5.4 presenta algunos ejemplos. El capítulo finaliza con el apartado 5.5 de conclusiones.

5.1. Servicio virtualizado de Misión Crítica sobre red B5G

El servicio MCPTT, dirigido a cuerpos de seguridad, bomberos, emergencias médicas y servicios de rescate, es el que mayor implantación tiene entre los sercicios de misión crítica impulsados por el 3GPP. Actualmente, el servicio MCPTT está soportado por un servidor centralizado. Siguiendo el estándar definido por el 3GPP [148], el servidor MCPTT tiene los roles de *MCPTT AS Participating* y *MCPTT AS Controlling*. Por tanto, todas las funcionalidades se proporcionan desde el servidor y todas las comunicaciones pasan por él. Los clientes interactúan con la función *participating* del servidor en todas las transacciones de la aplicación MCPTT.

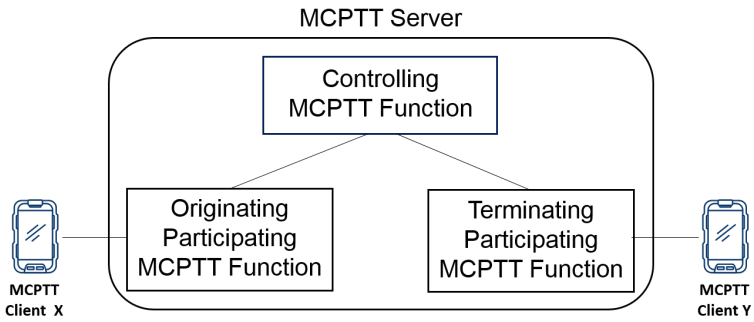


Figura 5.1: Funciones básicas MCPTT.

El hecho de que todas las comunicaciones, tanto del plano de control como del de usuario, pasen por el servidor puede conllevar, en determinadas condiciones de red, problemas de latencia y el consiguiente incumplimiento de indicadores KPI. Sin embargo, en las redes 5G y B5G, es posible adoptar una solución orientada hacia MEC, en la que algunas funcionalidades implementadas en VNF se llevan a los extremos de la red móvil. De esta forma, los servicios trasladados al extremo están más cerca de los usuarios y estos reciben el servicio con una latencia menor. Por tanto, parece lógico plantear un servicio virtualizado de Misión Crítica MCX (que incluye PTT, datos y vídeo) que permita realizar un despliegue inteligente de funciones VNF. En un escenario habitual de servicios de misión crítica, el servidor *MCPTT Application Server (MCPTT AS)* está instalado en un

datacenter, que puede ser del operador de red o del proveedor del servicio. En el caso de que esté en el datacenter del operador, el intercambio de los mensajes de señalización con el sistema IMS del operador puede justificar dicha posición centralizada del MCPTT AS. Por tanto, en ese caso, no parece beneficioso que todas las funcionalidades sean trasladados a los extremos de la red.

En este capítulo, se propone un modelo basado en teoría de colas que determina cómo realizar dicho despliegue de entidades VNF en función de la demanda de servicio, de la disponibilidad de recursos computacionales y de la latencia de servicio. La arquitectura de servicios de misión crítica que se pretende modelar en este capítulo consta de los siguientes elementos:

- Un servidor de aplicación de Misión Crítica, que se denominará *MCX Server*.
- Un nodo IMS del operador de red con el que el sistema MC intercambia mensajes de señalización.
- Un conjunto de nodos de acceso $\{N_1, N_2, N_3, \dots\}$, que tienen recursos computacionales para poder ejecutar funciones VNF.
- Un conjunto de usuarios $\{u_1, u_2, u_3, \dots\}$ que se conectan a la red por sus respectivos nodos de acceso.

Sobre esta arquitectura, se definen las caracteríticas que se detallan en los siguientes puntos.

5.1.1. Clases de tráfico

En primer lugar, se definen las siguientes clases o tipos de tráfico para el modelo:

T1- Tráfico de señalización. La provisión de servicios de Misión Crítica (MC) requiere un *SIP Core* tal como IMS. Las operadoras móviles ya tienen desplegado su sistema IMS como *core framework*. La comunicación del sistema MC con el *core* IMS solamente se produce durante las operaciones de control, por ejemplo, en el establecimiento de la comunicación. Esta clase contiene ese tipo de tráfico de señalización.

T2- Tráfico *floor-control*. Esta clase identifica al tráfico de control intercambiado para el arbitrio del medio. En una comunicación MCPTT es necesario gestionar en cada momento a quién corresponde el turno para hablar.

T3- Tráfico PTT (Push To Talk). Es el tráfico de voz de los usuarios del servicio MCPTT.

T4- Tráfico MCDATA. Son los datos de los usuarios del servicio MCDATA.

T5- Tráfico MCVIDEO. Es el tráfico de vídeo de los usuarios del servicio MCVIDEO.

En la clasificación que se ha hecho en cinco tipos de tráfico, se distinguen dos tipos de tráfico de control (los tipos T1 y T2) debido a que sus flujos no pasan por las mismas entidades software. Un ejemplo de tráfico tipo T1 es el proceso de establecimiento de una llamada entre dos clientes MCPTT. En este proceso, intervienen los clientes MCPTT, el servidor MCPTT y el sistema IMS del operador.

El cliente MCPTT X, que quiere iniciar una comunicación PTT, envía un mensaje de señalización SIP de tipo *INVITE* a su *MCPTT AS Originating Participating*. Este mensaje SIP, antes de llegar al *MCPTT AS Originating Participating*, pasa por el sistema IMS del operador. Primero llega al Proxy de IMS (P-CSCF) y este lo envía al Serving de IMS (S-CSFCS), para que descubra el *Public Service Identifier (PSI)* con el que direccionar al *MCPTT AS Originating Participating*. Este servidor reenvía un mensaje SIP *INVITE* al servidor *MCPTT AS Controlling* donde le indica el UE al que el cliente X está llamando. A continuación, el *MCPTT AS Controlling* reenvía el mensaje SIP *INVITE* al *MCPTT AS Terminating Participating* que será el encargado de llamar al cliente Y de supervisar toda la llamada. Sin embargo, para poder llegar al UE del cliente Y, previamente debe consultar en al módulo de IMS *I-CSFF* quien, a su vez, consulta el registro del cliente en la base de datos HSS y obtiene el S-CSCF que sirve al cliente Y. En este momento se reenvía el mensaje SIP al proxy IMS a través de C-CSCF. Si el cliente Y está configurado en modo *automatic commencement*, se generan automáticamente todas las confirmaciones en sentido inverso [99].

Una vez terminada la parte de señalización SIP del establecimiento de llamada, comienza la parte de *floor control* para determinar a quién corresponde el turno de hablar. Esta comunicación se lleva a cabo mediante protocolo RTCP y en ella solamente intervienen el cliente y el servidor MCPTT. Por tanto, a diferencia del tráfico tipo T1, este flujo de control no pasa por el sistema IMS del operador. Por ello, se establece que sea una clase diferente, concretamente, la tipo T2. Este *floor control* seguirá utilizándose a lo largo de toda la llamada, ya que es imprescindible tener controlado siempre el acceso al canal.

Posteriormente, cuando el cliente dispone del turno para hablar, envía los paquetes de voz PTT hacia su MCPTT AS Participating para que este los redirija hacia el destinatario (en caso de ser una llamada privada) o los destinatarios (en caso de ser una llamada de grupo) siguiendo el esquema *Originating Participating-Controlling- Terminating Participating* mostrado en la Figura 5.1. Para el envío de tráfico PTT, clasificado como tipo T3 en el modelo, se utiliza el protocolo RTP.

5.1.2. Funciones VNF

El modelo supone que es posible virtualizar el servicio de misión crítica separando sus funcionalidades en distintas VNF y que estas pueden ser desplegadas en diferentes puntos de la red. Para la construcción del modelo se definen las siguientes VNF:

- **VNF-1**. Este tipo de función procesa el tráfico de señalización IMS (tipo T1). En las implementaciones actuales, este tipo de procesamiento se lleva a cabo en el sistema IMS del operador de red y en el servidor de Misión Crítica (*MCX Server*). Por ello, se distinguirá una **VNF-1$^{\mathbf{IMS}}$** y una **VNF-1$^{\mathbf{MCX}}$**.
- **VNF-2**. Funciones que procesan tráfico de tipo *floor-control* (tipo T2).
- **VNF-3**. Función que procesa tráfico Push-To-Talk (tipo T3).
- **VNF-4**. Función que procesa tráfico de datos (tipo T4).
- **VNF-5**. Función que procesa tráfico de vídeo (tipo T5).
- **VNF-S** y **VNF-D**. En el paradigma de una red virtualizada, los servicios MCX se instalan junto a otros sobre la misma infraestructura física. Por tanto, en los nodos de acceso, la interfaz de radio y los recursos computacionales están compartidos entre diferentes *slices*. Para representar esta situación en el modelo, se definen: por un lado, las VNF-S para representar el consumo computacional de todo el tráfico *uplink* que recibe el nodo; por otro lado, las VNF-D para representar el consumo computacional de todo el tráfico *downlink* que sale del nodo hacia los usuarios conectados al mismo. En cada nodo de acceso hay, al menos, una VNF-S y una VNF-D.

Con la virtualización del servicio, se abre la opción de que estas VNF estén desplegadas en distintos puntos de la red. Algunas de ellas tendrán una ubicación asignada desde el principio, mientras que, para otras, se buscará la ubicación óptima con respecto al criterio de rendimiento que se establezca. Así, una de las funciones que ya está asignada es la VNF-1^{IMS} que se ejecuta en el nodo IMS. Ocurre lo mismo con las funciones VNF-S y VNF-D de los nodos de acceso; su ubicación es conocida desde el principio. La notación empleada para ellas será: VNF-S1/VNF-D1 para las funciones de este tipo instaladas en el nodo de acceso N_1, VNF-S2/VNF-D2 para las del nodo de acceso N_2 y así sucesivamente.

Tabla 5.1: Ubicaciones posibles para las funciones VNF.

Función	Tipo de ubicación	Posibles nodos
VNF-1^{IMS}	Predeterminada	Nodo IMS
VNF-1^{MCX}	No predeterminada	MCX Server, N_1, N_2, N_3...
VNF-2	No predeterminada	MCX Server, N_1, N_2, N_3...
VNF-3	No predeterminada	MCX Server, N_1, N_2, N_3...
VNF-4	No predeterminada	MCX Server, N_1, N_2, N_3...
VNF-5	No predeterminada	MCX Server, N_1, N_2, N_3...
VNF-S_i	Predeterminada	Nodo N_i
VNF-D_i	Predeterminada	Nodo N_i

Para el resto de funciones, se estudiará cuál es su ubicación ideal en función de las condiciones que se tengan en el sistema. En las implementaciones vistas hasta ahora, lo habitual es que todas esas funciones residan

en el servidor MCX. Sin embargo, con el modelo que se va a proponer, no tiene por qué ser así. La Tabla 5.1 resume las posibilidades que se tiene en cuanto a la ubicación de las VNF en los distintos nodos del escenario de estudio.

5.1.3. Cadenas de servicio

Si se considera que cada tipo de tráfico constituye un servicio, se puede definir una cadena de servicio para cada uno de ellos, esto es, la secuencia de funciones VNF que debe ejecutarse con cada clase de tráfico. Así, se tienen los siguientes casos:

- Cadena de servicio para el tráfico de clase T1 (señalización). Tal y como muestra la Figura 5.2 y como se ha explicado en el ejemplo de un establecimiento de llamada MCPTT, el procesamiento del flujo correspondiente a este servicio se inicia en las VNF-S de los nodos de acceso, sigue con la función VNF1 del nodo IMS, para pasar a continuación a la función VNF^{MCX} cuya respuesta se devuelve al sistema IMS y este la redirige, a través de la función VNF-D del nodo destino, al usuario correspondiente. Nótese que el nodo origen y el destino pueden ser iguales.

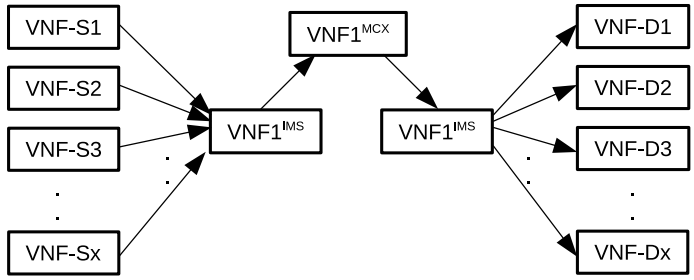


Figura 5.2: Cadena de servicio para el tráfico de clase T1.

- Cadena de servicio para el tráfico de clase T2 (*floor-control*). Tal y como muestra la Figura 5.3, el procesamiento del flujo correspondiente a este servicio se inicia en cualquiera de las funciones VNF-S de los nodos de acceso, sigue con la función VNF2 y finaliza en cualquiera de las funciones VNF-D de los nodos destino.

- Las cadenas de servicio correspondientes a los tráficos de clases T3, T4 y T5 (PTT, datos y vídeo respectivamente) son análogas a la del tráfico de clase T2.

Las cadenas de servicio presentadas en las Figuras 5.2-5.3 solamente tienen una instancia por cada VNF. Sin embargo, podría darse el caso de

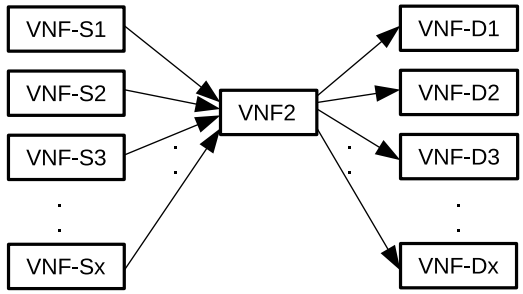


FIGURA 5.3: Cadena de servicio para el tráfico de clase T2.

que, en determinadas circunstancias (por ejemplo, que haya VNF saturadas), se creará más de una instancia de la misma VNF con objeto de distribuir la carga y obtener una mejora de rendimiento. Si se tiene esta configuración, es habitual que haya una función responsable de distribuir la carga (función *load balancer*). El modelo también estudiará dónde ubicar todas estas instancias de la VNF replicada, así como la función que distribuye la carga. La Figura 5.4 presenta un ejemplo donde hay varias instancias de la función VNF5 y una función *load balancer* representada como VNF5^{LB}.

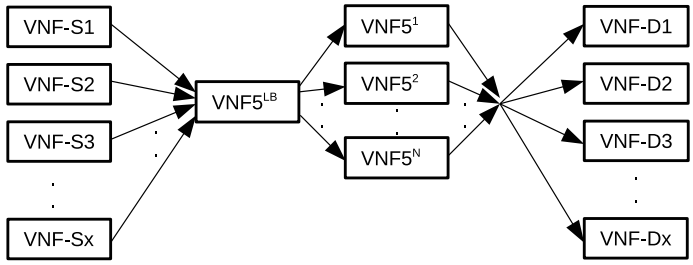


FIGURA 5.4: Ejemplo de cadena de servicio con varias instancias de la misma VNF.

5.1.4. Grupos

Como se ha mencionado ya, parte del tráfico puede corresponder a llamadas de grupo. Por ello, se tendrán definidos los grupos G_i con los usuarios u_j que pertenecen a cada uno de ellos. Por ejemplo, $G_1 = \{u_1, u_2, u_3, u_4\}$, $G_2 = \{u_1, u_5, u_6\}$ indica que el grupo denominado G_1 está compuesto por los usuarios u_1, u_2, u_3 y u_4, y que el grupo llamado G_2 lo forman u_1, u_5 y u_6. Tal y como se aprecia en el ejemplo, un usuario puede ser miembro de más de un grupo.

5.2. Modelo de red abierta de colas

En este apartado se propone un modelo para representar los recursos computacionales y de almacenamiento implicados en la provisión de un conjunto de servicios de Misión Crítica sobre una red B5G. Su objetivo es determinar cómo debe desplegarse un conjunto de funciones VNF para que se cumplan unos indicadores de rendimiento o KPIs. Para ello, junto con el modelo, se plantea un problema de optimización.

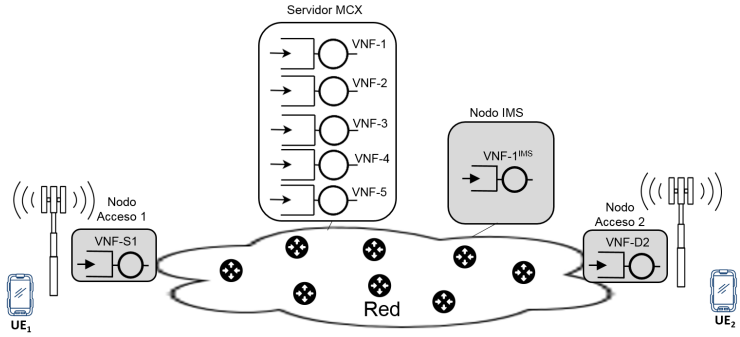


Figura 5.5: Ejemplo de distribución de VNF/colas.

En el modelo, cada función VNF será representada mediante una cola M/M/1. La Figura 5.5 muestra un ejemplo en el que todas las funciones del servicio MCX-PTT (VNF-1, VNF-2, VNF-3, VNF-4 y VNF-5) están alojadas en el servidor MCX; es decir, este caso es el escenario equivalente al de la implementación actual del servicio MCX-PTT, donde todas las funciones están centralizadas en el servidor y todas las comunicaciones pasan por él.

Además, aparece una cola en el nodo IMS (VNF-1^{IMS}) que representa el procesado de la señalización SIP que intercambian las entidades MCPTT y el sistema IMS del operador. Asimismo, en los nodos de acceso, también hay colas que representan el consumo de recursos computacionales.

Así, si el usuario UE_1 establece comunicación con el usuario UE_2 y transmite, por ejemplo, un mensaje de voz PTT (tráfico de tipo T3), su flujo de paquetes de voz, dentro del modelo, atravesará: primeramente, la VNF-S1 de su nodo de acceso; de este pasará al servidor MCX donde la VNF-3 lo procesará y lo reenviará hacia el nodo de acceso 2, ya que este es el nodo al que está conectado el usuario UE_2; el procesamiento en el nodo de acceso 2 está representado por la cola de la VNF-D2 y, finalmente, el los paquetes de voz serán retransmitidos para, finalmente, llegar al usuario destinatario.

5.2.1. Datos iniciales para el modelo

Antes de definir el modelo en detalle y abordar el problema de optimización, es necesario disponer de un conjunto de datos iniciales que son los

siguientes:

- Localización de los usuarios, esto es, por cuál de los nodos de acceso está conectado cada usuario.
- Configuración de grupos, es decir, cómo están definidos los grupos, indicando los miembros de cada uno.
- Cantidad de tráfico que genera cada usuario, especificando clase (de tráfico), destino y, en el caso de referirse a llamadas de grupo, cuál es el grupo al que corresponde.

Con estos datos iniciales, se puede contabilizar el tráfico que debe soportar cada VNF. También se puede estimar si alguna de las VNF está saturada o al borde de ello, por lo que se requieren instancias adicionales de esa VNF. No obstante, se podrá tomar la opción de crear réplicas de VNF, sin haber previsto situaciones de saturación.

5.2.2. Definición del modelo

El modelo propuesto es una red de colas M/M/1, donde cada cola q, perteneciente al conjunto $\mathcal{Q}$, representa a una VNF y los clientes de las colas corresponden a peticiones de procesamiento VNF. Cada tipo de tráfico (T1, T2, T3, T4 o T5) es una clase de cliente k perteneciente al conjunto $\mathcal{K}$. Todas las clases reciben el mismo tipo de servicio. Por tanto, cada cola q_i tiene una única tasa de servicio μ_i. Esto modela la situación en la que una misma instancia VNF atiende a peticiones de distintas clases de tráfico.

La tasa de servicio de la cola q, μ_q está directamente relacionada con el tiempo de ejecución de la función VNF. Dado que se suponen tiempos con distribución exponencial, si se denomina $\bar{t}_{vnf}$ al tiempo medio de ejecución de la VNF, entonces $\bar{t}_{vnf} = 1/\mu_q$. Por otro lado, se tiene un tiempo de ocupación de CPU por parte de la VNF. El tiempo de ejecución y el tiempo de ocupación de CPU no tienen por qué ser iguales, ya que, durante la ejecución de la función, puede haber tiempos de espera (por ejemplo, debidos al modo de funcionamiento de un protocolo) en los que el recurso computacional está dedicado a otras tareas; en cambio, dicho tiempo de espera sí que hay que considerarlo para el tiempo t_{vnf}. La ocupación de CPU y otros recursos está contabilizada con el parámetro v_q que representa el consumo computacional de la VNF representada en el modelo por la cola q.

Asimismo, se considera que cada VNF conlleva un consumo energético en el nodo en la que esté desplegada. El parámetro w_q indica el consumo energético asociado a la VNF representada en el modelo por la cola q.

Cada VNF o cola q tiene una tasa de llegada por cada clase k, denotada con λ_q^k (o λ_i^k en el caso de que la cola esté denotada con q_i). A su vez,

Tabla 5.2: Notación del modelo de colas para servicios MCX.

$\mathcal{K}$	Conjunto de clases de tráfico
$\gamma^k_{U\,a,b}$	Tráfico de usuario de clase k con origen a y destino b
$\mathcal{N}$	Conjunto de nodos de acceso y servidores
M	Número de nodos de acceso de ña red
$\gamma^k_{Nl,m}$	Tráfico agregado de clase k con origen el nodo l y destino el nodo m
$\mathcal{Q}$	Conjunto de colas que representan VNF
q_i	Cola que representa a la VNFi
q_{sx}	Cola que corresponde a la VNF-Sx desplegada en el nodo de acceso N_x
q_{dy}	Cola que corresponde a la VNF-Dy desplegada en el nodo de acceso N_y
q_1^{IMS}	Cola que representa a la VNF1$^{\text{IMS}}$
q_1^{MCX}	Cola que representa a la VNF1$^{\text{MCX}}$
μ_q	Tasa de servicio de la cola q
μ_i	Tasa de servicio de la cola q_i
μ_1^{IMS}	Tasa de servicio de la cola q_1^{IMS}
μ_1^{MCX}	Tasa de servicio de la cola q_1^{MCX}
λ_i	Tasa total de llegadas de peticiones de cualquier clase a la cola q_i
$A_{n,i}$	Variable binaria que indica si se despliega la cola q_i en el nodo n
$\delta_{l,m}$	Latencia entre el par de nodos l y m
R_i^k	Retardo medio de una petición de la clase k en la VNFi
D^k	Retardo medio total de una petición de la clase k
D_k^{KPI}	Retardo máximo permitido por el KPI asociado a la clase k
v_q	Consumo computacional de la VNF asociada a la cola q
v_i	Consumo computacional de la VNF asociada a la cola q_i
V_n	Capacidad computacional del nodo n
w_q	Consumo energético de la VNF asociada a la cola q
w_i	Consumo energético de la VNF asociada a la cola q_i
W_n	Máximo consumo energético permitido en el nodo n
p^k_{qr}	Probabilidad de que el flujo saliente de clase k de la cola q vaya a la cola r

cada cola q tiene una tasa total de llegadas denotada con λ_q (o λ_i) que es la suma de todas las tasas de llegada por clase.

$$\lambda_q = \sum_{k\in\mathcal{K}} \lambda_q^k, \quad \forall q \in \mathcal{Q} \qquad \text{o} \qquad \lambda_i = \sum_{k\in\mathcal{K}} \lambda_i^k, \quad \forall q_i \in \mathcal{Q} \tag{5.1}$$

Los nodos que intervienen en el modelo están representados por el conjunto $\mathcal{N} = \{N_{\text{IMS}}, N_{\text{MCX}}, N_1, N_2, N_3 \dots\}$, siendo N_{IMS} el nodo IMS, N_{MCX} el servidor MCX, y $N_1, N_2, N_3 \dots$ los distintos nodos de acceso a la red. Cada nodo $n \in \mathcal{N}$ tiene una capacidad computacional V_n disponible para las VNF que se instalen en él. Asimismo, en cada nodo hay una cantidad máxima de consumo energético permitido, W_n, para el conjunto de VNF instaladas.

Los nodos están conectados a través de la red. Esta conexión puede ser a través de un enlace directo, que no es lo habitual, o a través de una serie de enlaces y nodos intermedios de la red. Si existe el enlace físico directo entre los nodos n_l y n_m, se tendrá el dato de la capacidad $C_{l,m}$ de dicho enlace. Si la conexión es a través de múltiples enlaces, estos son considerados como un único enlace virtual cuyo valor $C_{l,m}$ corresponde al del enlace con menor capacidad. Hecha esta abstracción, en cualquiera de los casos, se supone que se conoce la latencia de red $\delta_{l,m}$ entre cada par de nodos n_l, n_m.

5.3. Formulación del problema

El problema que se pretende resolver consiste en determinar cómo debe realizarse el despliegue de VNF teniendo el retardo de servicio como parámetro principal o métrica básica de rendimiento. Para ello se plantea una función objetivo, que busca minimizar el retardo, y una serie de restricciones que es preciso cumplir.

5.3.1. Variable de decisión

Se define la variable de decisión $A_{n,q} \in \{0,1\}$ que indica si la cola q está desplegada en el nodo n o no. En caso afirmativo $A_{n,q} = 1$; en caso negativo $A_{n,q} = 0$. Se debe cumplir la igualdad:

$$\sum_{n \in \mathcal{N}} A_{n,q} = 1, \quad \forall q \in \mathcal{Q}. \tag{5.2}$$

5.3.2. Restricciones del sistema

En primer lugar, para asegurar la estabilidad del conjunto de colas M/M/1 que conforman el sistema, en cada cola, cada tasa total de llegadas debe ser menor que la tasa de servicio, es decir,

$$\lambda_i < \mu_i, \quad \forall q_i \in \mathcal{Q}. \tag{5.3}$$

Tanto λ_i como μ_i son parámetros de entrada del modelo que se pueden calcular con los datos iniciales. Si no se cumple la condición de estabilidad de la Ecuación 5.3 para la cola q_i, se deben crear instancias adicionales para dicha VNF, tantas como sean necesarias para cumplir la condición de estabilidad. En ese caso, las múltiples instancias de la cola q_i se denotarán con $q_{i1}, q_{i2}, q_{i3} \ldots$ y habrá que recalcular las tasas de entrada $\lambda_{i1}, \lambda_{i2}, \lambda_{i3} \ldots$ en función de la distribución de tráfico que realice la VNF^{LB} que distribuye la carga entre las diferentes instancias replicadas.

Otra de las restricciones es que no se puede exceder la capacidad computacional de cada nodo:

$$\sum_{q \in \mathcal{Q}} A_{n,q} v_q \leq V_n, \quad \forall n \in \mathcal{N}. \tag{5.4}$$

De igual forma, tampoco se puede exceder el máximo consumo energético permitido en cada nodo:

$$\sum_{q \in \mathcal{Q}} A_{n,q} w_q \leq W_n, \quad \forall n \in \mathcal{N}. \tag{5.5}$$

Por último, las capacidades $C_{l,m}$ limitarán el tráfico que se puede cursar entre las VNF desplegadas en el nodo l y las instaladas en el nodo m.

$$\sum_{k \in \mathcal{K}} \sum_{q,r \in \mathcal{Q}} \lambda_q^k p_{qr}^k A_{l,q} A_{m,r} \leq C_{l,m}, \tag{5.6}$$

siendo p_{qr}^k la probabilidad de que el flujo saliente de clase k de la cola q, que se encuentra en el nodo l, vaya a la cola r, que está instalada en el nodo m.

5.3.3. Retardos de servicio e indicadores KPI

Como se ha mencionado anteriormente, la función objetivo que se plantea está relacionada con el retardo del servicio, concretamente, con la combinación de tiempos de ejecución de VNF y latencias de red requerida para completar las cadenas de servicio. El objetivo es minimizar esas latencias de servicio cumpliendo, al mismo tiempo, tanto los KPI exigidos como las restricciones del apartado anterior. Para ello, se plantea el parámetro D_k, que se define como el retardo medio de una petición de clase k en completar su cadena de servicio. Dentro de D_k, se distinguen dos partes:

- Los retardos asociados a las VNF, es decir, el tiempo que una petición requiere para atravesar cada cola q que representa una VNF de su cadena de servicio. Dado que el modelo está constituido por colas M/M/1 [149], el retardo de una petición de clase k en la cola q_i será:

$$R_i^k = \frac{1}{\mu_i - \lambda_i}, \quad \forall q_i \in \mathcal{Q}. \tag{5.7}$$

  La expresión de R_i^k no depende de la clase k, ya que se ha supuesto que el tipo de servicio es el mismo para todas las clases.

- Las latencias de red en las conexiones entre diferentes nodos.

Por tanto, se puede expresar D_k como:

$$D_k = T_{\text{vnfs}}^k + T_{\text{red}}^k, \tag{5.8}$$

donde T_{vnfs}^k será función de R_i^k y T_{red}^k dependerá de las latencias de red $\delta_{l,m}$ entre cada par de nodos.

A continuación, se calcula el retardo medio asociado a cada cadena de servicio. Se considera que el número de nodos de acceso de la red es M.

5.3.3.1. Clase de tráfico k=1

En el caso de la clase k=1, hay que tener en cuenta que este tipo de tráfico pasa por el sistema IMS del operador, pues es necesario intercambiar señalización con el *SIP Core*. El retardo medio D_1 se calcula de la siguiente manera:

$$D_1 = T_{\text{vnfs}}^1 + T_{\text{red}}^1, \tag{5.9}$$

La expresión de T_{vnfs}^1 tiene en cuenta que, en esta cadena de servicio, se pasa dos veces por el sistema IMS del operador, porque se realizan dos consultas SIP. Obviamente, también se considera el procesamiento de las VNF de los nodos de acceso y de la VNF-1^{MCX}, que puede ubicarse en el

servidor MCX pero también en cualquiera de los nodos de acceso.

$$T^1_{\text{vnfs}} = \sum_{l=1}^{M}\sum_{m=1}^{M} p_{l,m}R_{Sl} + R_1^{\text{IMS}} + R_1^{\text{MCX}} + R_1^{\text{IMS}} + \sum_{m=1}^{M}\sum_{l=1}^{M} p_{l,m}R_{Dm}, \quad (5.10)$$

siendo $p_{l,m} = \frac{\gamma^1_{Nl,m}}{\gamma^1_T}$ con $\gamma^1_T = \sum_{l=1}^{M}\sum_{m=1}^{M} \gamma^1_{Nl,m}$.

La expresión de T^1_{red} queda en función de los tiempos de latencia $\delta_{l,m}$ y de las variables de decisión $A_{n,i}$:

$$\begin{aligned} T^1_{\text{red}} = &\sum_{l=1}^{M}\sum_{y\in\mathcal{N}} p_l A_{(Ny,1\text{ims})}\delta_{l,y} + \sum_{x,y\in\mathcal{N}} A_{(Nx,1\text{ims})}A_{(Ny,1\text{mcx})}\delta_{x,y} + \\ &+ \sum_{x,y\in\mathcal{N}} A_{(Nx,1\text{mcx})}A_{(Ny,1\text{ims})}\delta_{x,y} + \sum_{m=1}^{M}\sum_{x\in\mathcal{N}} p_m A_{(Nx,1\text{mcx})}\delta_{x,m}, \end{aligned} \quad (5.11)$$

siendo $p_l = \frac{\sum_{m=1}^{M}\gamma^1_{Nl,m}}{\gamma^1_T}$ y $p_m = \frac{\sum_{l=1}^{M}\gamma^1_{Nl,m}}{\gamma^1_T}$.

5.3.3.2. Clases de tráfico k=2,3,4,5

La forma de las cadenas de servicio para las clases $k = 2, 3, 4, 5$ (en el caso de que solo haya una instancia por cada VNF) son iguales, por lo que las expresiones de sus retardos medios de servicio pueden agruparse en las siguientes expresiones:

$$D_k = T^k_{\text{vnfs}} + T^k_{\text{red}}, \quad k = 2, 3, 4, 5. \quad (5.12)$$

La expresión de T^k_{vnfs} con k=2,3,4,5 queda de la siguiente manera:

$$T^k_{\text{vnfs}} = \sum_{l=1}^{M}\sum_{m=1}^{M} p_{l,m}R_{Sl} + R_k + \sum_{m=1}^{M}\sum_{l=1}^{M} p_{l,m}R_{Dm}, \quad (5.13)$$

siendo $p_{l,m} = \frac{\gamma^k_{Nl,m}}{\gamma^k_T}$ con $\gamma^k_T = \sum_{l=1}^{M}\sum_{m=1}^{M} \gamma^k_{Nl,m}$.

La expresión de T^k_{red} con k=2,3,4,5 queda así:

$$T^k_{\text{red}} = \sum_{l=1}^{M}\sum_{y\in\mathcal{N}} p_l A_{(Ny,k)}\delta_{l,y} + \sum_{m=1}^{M}\sum_{x\in\mathcal{N}} p_m A_{(Nx,k)}\delta_{x,m}, \quad (5.14)$$

siendo $p_l = \frac{\sum_{m=1}^{M}\gamma^k_{Nl,m}}{\gamma^k_T}$ y $p_m = \frac{\sum_{l=1}^{M}\gamma^k_{Nl,m}}{\gamma^k_T}$.

5.3.4. Función objetivo

Como se ha mencionado anteriormente, la función objetivo que se plantea está relacionada con el retardo del servicio, concretamente, con la combinación de tiempos de ejecución de VNF y latencias de red requerida para completar las cadenas de servicio. El objetivo es minimizar esos tiempos de servicio cumpliendo, al mismo tiempo, tanto los KPI exigidos como las restricciones del apartado anterior.

$$\arg\min_{A} \left(\max_{k \in \mathcal{K}} \frac{D_k}{D_k^{\text{KPI}}} \right) \tag{5.15}$$

5.3.5. Método de resolución

Para resolver el problema de optimización, se sigue el método empleado en [112], adaptándolo a las particularidades del entorno de servicios Mission-Critical Services (MCX). Por ello, se procede con los siguientes pasos:

1) Plantear una versión convexa del problema.
2) Resolver el problema convexo con un *solver*.
3) Utilizar la heurística denominada *MaxZ* para estimar que la cola identificada como q^* se debe desplegar en el nodo identificado como n^*.
4) Repetir los pasos 1-3 hasta que se determine el despliegue de todas las VNF.

Para llevar a cabo el primer paso, el planteamiento del problema convexo, se sustituyen las variables binarias $A_{n,q}$ por variables continuas $\tilde{A}_{n,q} \in [0,1]$. También es necesario sustituir los productos de las variables $\tilde{A}$ por una nueva variable $\Phi_{(l,q,m,r)}$ que cumple las siguientes condiciones:

$$\Phi_{(l,q,m,r)} \leq \tilde{A}_{l,q}, \quad \forall l,m \in \mathcal{N}, \quad \forall q,r \in \mathcal{Q}; \tag{5.16}$$

$$\Phi_{(l,q,m,r)} \leq \tilde{A}_{m,r}, \quad \forall l,m \in \mathcal{N}, \quad \forall q,r \in \mathcal{Q}; \tag{5.17}$$

$$\Phi_{(l,q,m,r)} \geq \tilde{A}_{l,q} + \tilde{A}_{m,r} - 1, \quad \forall l,m \in \mathcal{N}, \quad \forall q,r \in \mathcal{Q}; \tag{5.18}$$

En el segundo paso, se utiliza un *solver* que recibe las siguientes expresiones:

⇒ Función objetivo 5.15 (introduciendo en ella las ecuaciones 5.8 - 5.14)

⇒ Sujeto a restricciones:

- Ecuación $\sum \tilde{A}_{n,q} = 1$
- Ecuaciones 5.3 - 5.6
- Ecuaciones 5.16 - 5.18

Después de aplicar el segundo paso, el *solver* da como resultado unos valores para las variables continuas $\tilde{A}_{n,q}$.

A continuación, el tercer paso, consta de dos partes:

- Determinar el denominado *Z-score* con la siguiente expresión:

$$Z_{n,q} = \tilde{A}_{n,q} + \mathbf{1}_{A_{n,q}\mu_q \geq T_{n,q}}. \tag{5.19}$$

  donde **1** es la función indicador.

- Realizar la asignación entre el nodo n^* y la VNF q^* con el siguiente criterio:

$$n^*, q^* \leftarrow \underset{n\in\mathcal{N}, q\in\mathcal{Q}}{\arg\max}\, Z_{n,q}. \tag{5.20}$$

5.4. Casos de estudio

Para demostrar la utilidad del modelo propuesto, en este apartado se presentan algunos ejemplos donde se resuelve numéricamente el modelo y se interpretan los resultados obtenidos. Se plantean casos donde se verá cómo afecta la localización de usuarios y la capacidad computacional disponible en los nodos. La Figura 5.6 muestra la arquitectura simplificada que se utilizará para los ejemplos. Como se puede apreciar, solamente tiene un nodo IMS, un servidor MCX y dos nodos de acceso N_1 y N_2.

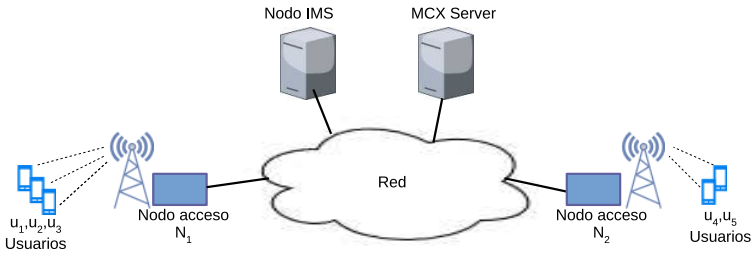


FIGURA 5.6: Arquitectura simplificada de servicios de Misión Crítica.

En todos los casos, el modelo dispondrá de los siguientes datos de entrada:

- Tiempo medio de ejecución de VNF ($\bar{t}_{vnf}$). Estos tiempos pueden ser extraídos del trabajo de tesis [99] donde se hace un análisis de trazas de los diversos protocolos utilizados en MCPTT. Con este dato, se establece el valor de cada μ_i del modelo.

- Latencia de red $\delta_{l,m}$ entre cada par de nodos.

- Localización de cada usuario, es decir, el nodo por el que accede cada usuario a la red.

- Tráfico generado por cada usuario, destino de cada tipo de tráfico y configuración de grupos para llamadas grupales. Con todos estos datos, el modelo conoce el tráfico agregado que atraviesa la red y, aplicando la topología de las cadenas de servicio, cómo se distribuye por los diferentes enlaces y VNF. Por tanto, ya se tiene el valor del parámetro λ_i de cada VNF.

- Conocidos μ_i y λ_i, se define el parámetro ρ_i de cada VNF como $\rho_i = \lambda_i/\mu_i$. Da una idea del grado de saturación de cada VNF desde el punto de vista computacional.

Por motivos de simplicidad, solamente habrá tres tipos de tráfico: señalización, *floor control* y PTT. Por tanto, las VNF a desplegar en la red serán VNF-1, VNF-2 y VNF-3. Asimismo, indicar que, en los ejemplos, a pesar de que el 3GPP tiene definidos varios indicadores de rendimiento KPI de MCPTT sobre LTE [150] se utilizan valores no reales de indicadores KPI. No obstante, esto no afecta a las conclusiones que se derivan de la evaluación del modelo.

5.4.1. Caso 1

En el primero de los casos que se plantea, las latencias de red entre nodos son: $\delta_{\mathrm{MCX,IMS}}$=1 ms; $\delta_{\mathrm{N1,MCX}}$=2 ms; $\delta_{\mathrm{N1,IMS}}$=2 ms; $\delta_{\mathrm{N2,MCX}}$=2 ms; $\delta_{\mathrm{N2,IMS}}$=2 ms; $\delta_{\mathrm{N1,N2}}$=6 ms. Además, se supone que se tiene una carga de tráfico moderada, siendo $\rho_i = 0,1$ y $\bar{t}_{vnf} = 0,5$ ms en todas las VNF. Se va a distinguir un caso 1A y un caso 1B.

Caso 1A: Los usuarios se distribuyen por igual entre los dos nodos de acceso: 50 % de usuarios conectados al nodo N_1 y 50 % a N_2. Las probabilidades de transmisión de flujos entre los usuarios conectados a los nodos N_1 y N_2 son: $p_{11} = p_{12} = p_{21} = p_{22} = 0,25$.

La Figura 5.7 muestra el resultado que da el *solver* sobre las variables $\tilde{A}$ en cada una de las tres iteraciones que son necesarias para ubicar las VNF-1 (representada por q_1^{MCX}), VNF-2 (q_2) y VNF-3 (q_3). En este caso simple, viendo el máximo valor que sale en la matriz $\tilde{A}$, es posible determinar la asignación de cada iteración. Así, en este caso 1A, en la primera iteración, se asigna la cola q_1^{MCX} al servidor MCX, en la segunda iteración se asigna la cola q_2 al servidor MCX y, en la tercera iteración, q_3 al servidor MCX.

Por tanto, el despliegue resultante de las iteraciones queda así:

$$q_1^{\mathrm{MCX}} \rightarrow \mathrm{MCX}; \qquad q_2 \rightarrow \mathrm{MCX}; \qquad q_3 \rightarrow \mathrm{MCX}.$$

Y los siguientes retardos medios para cada clase de tráfico:
$D_1 = 8,7778$ ms.
$D_2 = 5,6667$ ms.
$D_3 = 5,6667$ ms.

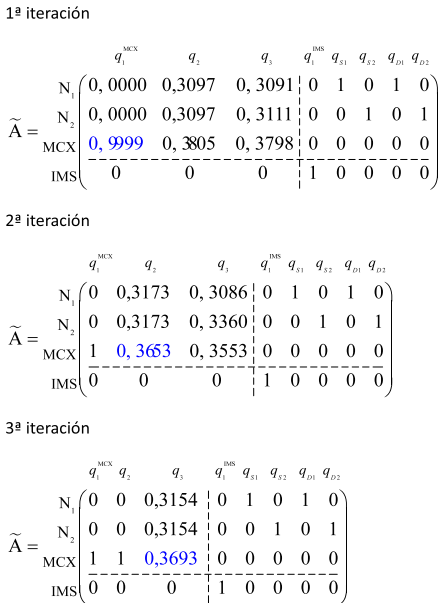


FIGURA 5.7: Resultados intermedios de las iteraciones para el caso 1A.

Caso 1B: Los usuarios no se distribuyen por igual entre los dos nodos de acceso: 70 % de usuarios conectados al nodo N_1 y 30 % a N_2. Las probabilidades de transmisión de flujos son: $p_{11} = 0,5$; $p_{12} = p_{21} = 0,2$; $p_{22} = 0,1$. El resto de datos son iguales que los del caso 1A.

El desarrollo de las iteraciones que se puede ver en la Figura 5.8 dice que, en la primera iteración, se asigna la cola q_1^{MCX} al servidor MCX, en la segunda iteración, se asigna la cola q_2 al nodo N_1, y en la tercera iteración, q_3 al nodo N_1

Por tanto, el despliegue resultante de las iteraciones queda así:

$$q_1^{\text{MCX}} \rightarrow \text{MCX}; \qquad q_2 \rightarrow N_1; \qquad q_3 \rightarrow N_1.$$

Y los siguientes retardos medios para cada clase de tráfico
$D_1 = 8,7778$ ms.
$D_2 = 5,2667$ ms.
$D_3 = 5,2667$ ms.

El caso 1 muestra que, dependiendo de dónde estén conectados los usuarios, el despliegue óptimo de funciones VNF es distinto. Así, mientras que en el caso 1A, la centralización de todos los servicios es la opción más apropiada, en el caso 1B la solución consiste en distribuir, por un lado, la VNF-1 en el servidor central y, por otro lado, las VNF-2 y VNF-3 en el nodo N_1.

1ª iteración

$$\widetilde{A} = \begin{array}{c} \\ N_1 \\ N_2 \\ MCX \\ IMS \end{array} \begin{array}{c} \begin{array}{ccc|ccccc} q_1^{MCX} & q_2 & q_3 & q_1^{IMS} & q_{S1} & q_{S2} & q_{D1} & q_{D2} \end{array} \\ \left(\begin{array}{ccc|ccccc} 0,0001 & 0.6960 & 0{,}8735 & 0 & 1 & 0 & 1 & 0 \\ 0,0001 & 0,0213 & 0,0062 & 0 & 0 & 1 & 0 & 1 \\ 0,9997 & 0,2827 & 0{,}1203 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{array}\right) \end{array}$$

2ª iteración

$$\widetilde{A} = \begin{array}{c} \\ N_1 \\ N_2 \\ MCX \\ IMS \end{array} \begin{array}{c} \begin{array}{ccc|ccccc} q_1^{MCX} & q_2 & q_3 & q_1^{IMS} & q_{S1} & q_{S2} & q_{D1} & q_{D2} \end{array} \\ \left(\begin{array}{ccc|ccccc} 0 & 0{,}8658 & 0,6851 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0,0054 & 0,0199 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0{,}1288 & 0,2851 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{array}\right) \end{array}$$

3ª iteración

$$\widetilde{A} = \begin{array}{c} \\ N_1 \\ N_2 \\ MCX \\ IMS \end{array} \begin{array}{c} \begin{array}{ccc|ccccc} q_1^{MCX} & q_2 & q_3 & q_1^{IMS} & q_{S1} & q_{S2} & q_{D1} & q_{D2} \end{array} \\ \left(\begin{array}{ccc|ccccc} 0 & 1 & 0{,}7853 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0,0288 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0{,}1860 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{array}\right) \end{array}$$

FIGURA 5.8: Resultados intermedios de las iteraciones para el caso 1B.

5.4.2. Caso 2

En el caso 2 se mantienen todos los datos del caso 1, salvo la latencia entre los nodos N_1 y N_2 que disminuye y pasa a ser $\delta_{N1,N2}$=3 ms.

Caso 2A: Los usuarios se distribuyen por igual entre los dos nodos de acceso: 50 % de usuarios conectados al nodo N_1 y 50 % a N_2. Las probabilidades de transmisión de flujos son: $p_{11} = p_{12} = p_{21} = p_{22} = 0,25$.

El desarrollo de las iteraciones de la Figura 5.9 muestra que, en la primera iteración, se asigna la cola q_1^{MCX} al servidor MCX, en la segunda iteración, se asigna la cola q_2 al nodo N_1, y en la tercera iteración, q_3 al nodo N_1. Cabe mencionar que en este caso, debido a la simetría total existente, se pueden hacer las asignaciones indistintamente a N_1 y a N_2.

El despliegue resultante de las iteraciones queda así:

$$q_1^{MCX} \rightarrow MCX; \qquad q_2 \rightarrow N_1; \qquad q_3 \rightarrow N_1.$$

Y los siguientes retardos medios para cada clase de tráfico
$D_1 = 8,7778$ ms.
$D_2 = 4,6667$ ms.
$D_3 = 4,6667$ ms.

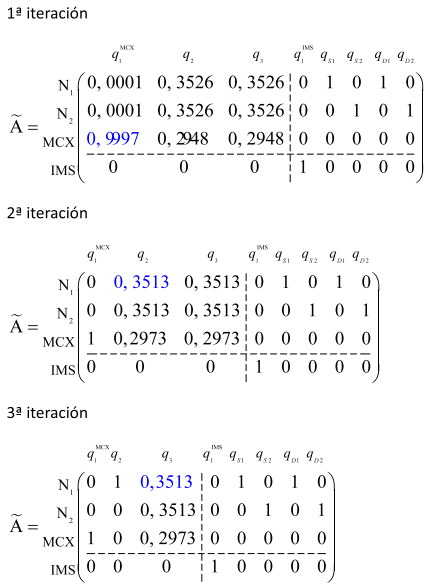


FIGURA 5.9: Resultados intermedios de las iteraciones para el caso 2A.

Caso 2B: Los usuarios no se distribuyen por igual entre los dos nodos de acceso: 70 % de usuarios conectados al nodo N_1 y 30 % a N_2. Las probabilidades de transmisión de flujos son: $p_{11} = 0,5$; $p_{12} = p_{21} = 0,2$; $p_{22} = 0,1$.

El despliegue resultante de las iteraciones que se pueden ver en la Figura 5.10 queda así:

$$q_1^{\text{MCX}} \rightarrow \text{MCX}; \qquad q_2 \rightarrow \text{N}_1; \qquad q_3 \rightarrow \text{N}_1.$$

Y los siguientes retardos medios para cada clase de tráfico
$D_1 = 8,7778$ ms.
$D_2 = 3,4667$ ms.
$D_3 = 3,4667$ ms.

En el caso 2, se observa que, para la VNF-1, siempre resulta mejor mantenerla en el servidor central. Además, tener una latencia baja entre los nodos N_1 y N_2 favorece el uso de la conexión "directa" entre ellos. Esto conlleva que el despliegue óptimo consista en acercar las VNF-2 y VNF-3 a los nodos donde más usuarios hay.

5.4.3. Caso 3

Aquí se plantea el caso en el que una VNF puede tener varias instancias. Las condiciones de partida son las mismas que las del caso 2A y se supone que hay un incremento del flujo de voz MCPTT (tráfico de clase 3). Los

1ª iteración

$$\widetilde{A} = \begin{array}{c} \\ N_1 \\ N_2 \\ MCX \\ IMS \end{array} \begin{array}{c} \begin{array}{ccccccccc} q_1^{MCX} & & q_2 & & q_3 & q_1^{IMS} & q_{S1} & q_{S2} & q_{D1} & q_{D2} \end{array} \\ \left(\begin{array}{ccc|ccccc} 0,0001 & & 0,3544 & & 0,3544 & 0 & 1 & 0 & 1 & 0 \\ 0,0001 & & 0,3544 & & 0,3544 & 0 & 0 & 1 & 0 & 1 \\ 0,9997 & & 0,2913 & & 0,2913 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & & 0 & & 0 & 1 & 0 & 0 & 0 & 0 \end{array}\right) \end{array}$$

2ª iteración

$$\widetilde{A} = \begin{array}{c} \\ N_1 \\ N_2 \\ MCX \\ IMS \end{array} \begin{array}{c} \begin{array}{ccccccccc} q_1^{MCX} & & q_2 & & q_3 & q_1^{IMS} & q_{S1} & q_{S2} & q_{D1} & q_{D2} \end{array} \\ \left(\begin{array}{ccc|ccccc} 0 & & 0,3435 & & 0,3435 & 0 & 1 & 0 & 1 & 0 \\ 0 & & 0,3211 & & 0,3211 & 0 & 0 & 1 & 0 & 1 \\ 1 & & 0,3354 & & 0,3354 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & & 0 & & 0 & 1 & 0 & 0 & 0 & 0 \end{array}\right) \end{array}$$

3ª iteración

$$\widetilde{A} = \begin{array}{c} \\ N_1 \\ N_2 \\ MCX \\ IMS \end{array} \begin{array}{c} \begin{array}{ccccccccc} q_1^{MCX} & q_2 & & q_3 & q_1^{IMS} & q_{S1} & q_{S2} & q_{D1} & q_{D2} \end{array} \\ \left(\begin{array}{cc|c|ccccc} 0 & 1 & 0,3435 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0,3211 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0,3354 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{array}\right) \end{array}$$

FIGURA 5.10: Resultados intermedios de las iteraciones para el caso 2B.

tráficos de las clases 1 y 2 siguen igual, por lo que el despliegue de VNF-1 y VNF-2 se mantiene como en el caso 2A. Por ello, aquí solamente se analiza la ubicación de la VNF-3.

El incremento del flujo MCPTT hace que aumente el parámetro λ_i de las colas que atraviesan ese tráfico de clase 3. Se observa a través del parámetro ρ_i, cómo afecta su variación en la latencia del servicio. Otra suposición que se hace es que, en este Caso 3, el servidor MCX dispone de más recursos computacionales que los nodos de acceso. Por ello, en el servidor se pueden desplegar dos instancias de la misma VNF, mientras que en los nodos de acceso no admiten más que una instancia por VNF. Con todos estos supuestos, se analiza cuál es la mejor ubicación para la VNF-3 a medida que aumenta el parámetro ρ_i. El hecho de que el servidor MCX puede tener dos instancias de la misma VNF hace que aumente su μ_i y por tanto disminuya su ρ_i. Supóngase que $\rho_3^{\mathrm{MCX}} = \rho_3^{\mathrm{N1}}/2$ y que $\rho_3^{\mathrm{N1}} = \rho_3^{\mathrm{N2}}$.

El despliegue resultante de las iteraciones que se muestran en la Figura 5.11 queda así:

$$\text{Con } \rho_3^{\mathrm{N1}} = 0,2\text{:} \quad q_1^{\mathrm{MCX}} \to \mathrm{MCX}; \qquad q_2 \to \mathrm{MCX}; \qquad q_3 \to \mathrm{N}_1.$$

$$\text{Con } \rho_3^{\mathrm{N1}} = 0,5\text{:} \quad q_1^{\mathrm{MCX}} \to \mathrm{MCX}; \qquad q_2 \to \mathrm{MCX}; \qquad q_3 \to \mathrm{N}_1.$$

$$\text{Con } \rho_3^{\mathrm{N1}} = 0,8\text{:} \quad q_1^{\mathrm{MCX}} \to \mathrm{MCX}; \qquad q_2 \to \mathrm{MCX}; \qquad q_3 \to \mathrm{MCX}.$$

Y los siguientes retardos medios para la clase de tráfico 3:
Con $\rho_3 = 0,2 \to D_3 = 4,8750$ ms.
Con $\rho_3 = 0,5 \to D_3 = 6,0000$ ms.

3ª iteración

$$\rho_3 = 0{,}2 \;;\; \rho_3^{MCX} = 0{,}1$$

$$\widetilde{A} = \begin{array}{c} \\ N_1 \\ N_2 \\ MCX^2 \\ IMS \end{array} \begin{array}{c} \begin{array}{ccccccccc} q_1^{MCX} & q_2 & & q_3 & & q_1^{IMS} & q_{S1} & q_{S2} & q_{D1} & q_{D2} \end{array} \\ \left(\begin{array}{ccc|ccccc} 1 & 1 & 0,3693 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0,3693 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0,2613 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{array}\right) \end{array}$$

3ª iteración

$$\rho_3 = 0{,}5 \;;\; \rho_3^{MCX} = 0{,}25$$

$$\widetilde{A} = \begin{array}{c} \\ N_1 \\ N_2 \\ MCX^2 \\ IMS \end{array} \begin{array}{c} \begin{array}{ccccccccc} q_1^{MCX} & q_2 & & q_3 & & q_1^{IMS} & q_{S1} & q_{S2} & q_{D1} & q_{D2} \end{array} \\ \left(\begin{array}{ccc|ccccc} 1 & 1 & 0,3436 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0,3436 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0,3127 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{array}\right) \end{array}$$

3ª iteración

$$\rho_3 = 0{,}8 \;;\; \rho_3^{MCX} = 0{,}4$$

$$\widetilde{A} = \begin{array}{c} \\ N_1 \\ N_2 \\ MCX^2 \\ IMS \end{array} \begin{array}{c} \begin{array}{ccccccccc} q_1^{MCX} & q_2 & & q_3 & & q_1^{IMS} & q_{S1} & q_{S2} & q_{D1} & q_{D2} \end{array} \\ \left(\begin{array}{ccc|ccccc} 1 & 1 & 0,1413 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0,1413 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0,7173 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{array}\right) \end{array}$$

FIGURA 5.11: Resultados intermedios de las iteraciones para el caso 3.

Con $\rho_3 = 0,8 \rightarrow D_3 = 9,8333$ ms.

En el Caso 3 se observa que, con intensidades bajas de tráfico MCPTT, es mejor desplegar una sola VNF en el nodo N_1, acercando el servicio a los usuarios conectados a dicho nodo. No compensa utilizar la mayor capacidad computacional del servidor central. En cambio, con intensidad de tráfico alta, los tiempos de respuesta de las VNF se ven afectados y, entonces, sí resulta ventajoso utilizar la capacidad computacional del servidor central para desplegar en él varias instancias de la misma VNF.

5.5. Conclusiones

En este capítulo se ha presentado un modelo basado en teoría de colas aplicado a un servicio virtualizado de Misión Crítica sobre red B5G. Esto satisface la idea de transferir conocimiento adquirido en el modelado de

sondas de análisis de tráfico en esta otra línea de investigación aplicada sobre servicios MCX-PTT.

Tras haber identificado los componentes (clases de tráfico, funciones VNF, cadenas de servicio) que es preciso incluir en el modelo, se ha propuesto un modelo de red abierta de colas M/M/1. Se ha desarrollado la formulación de la red de colas, para obtener la expresión del parámetro de rendimiento clave que, en este caso, es la latencia de cada tipo de tráfico identificado. A continuación se ha planteado un problema de optimización con una función objetivo que minimiza las latencias y una serie de restricciones relacionadas con la capacidad computacional y el coste energético que el despliegue de funciones VNF tiene en los nodos. Se ha resuelto el problema de optimización aplicando un método iterativo en el que se usa un *solver* y la heurística denominada MaxZ. De esta forma, se llega al resultado final que determina cómo deben desplegarse las VNF en los nodos y servidores. Para finalizar, se han presentado unos ejemplos teóricos donde se muestra la utilidad del modelo. En ellos, se aprecia que separar las funcionalidades del sistema MCPTT en funciones VNF y desplegarlas por diferentes partes de la red puede mejorar la calidad del servicio.

Esta propuesta de modelo no es más que un primer paso en el diseño de una nueva implementación de los servicios de Misión Crítica, donde podría separarse las funcionalidades del plano de usuario por un lado y las del plano de control por otro. Sin embargo, para ello, es necesario una estandarización que defina todos los protocolos requeridos para proveer el servicio basándose en un despliegue inteligente de funciones VNF. Una vez hecho eso, la siguiente tarea será desarrollar las VNF que cumplen ese estándar y proporcionan el servicio.

Capítulo 6

Conclusiones y líneas futuras

En este capítulo se presentan las conclusiones más importantes de esta Tesis Doctoral, a la vez que se destacan las aportaciones más interesantes y se proponen líneas futuras de trabajo.

El apartado 6.1 resume el trabajo desarrollado a lo largo de todo el proceso de investigación. A continuación, en el apartado 6.2, se describen las aportaciones más importantes que se extraen de la labor realizada en la tesis. Posteriormente, el apartado 6.3 expone las posibles vías de investigación que se plantean como continuación de esta tesis. Para terminar, el apartado 6.4 contiene las conclusiones finales y, en el apartado 6.5, se presentan las contribuciones realizadas para la divulgación de los resultados de esta investigación, tanto en publicaciones como en artículos de congresos nacionales e internacionales.

6.1. Resumen del trabajo realizado

Esta tesis presenta tres modelos basados en teoría de colas que, tomando como referencia los consumos computacionales de los elementos software que representan, miden el rendimiento del sistema bajo estudio.

En el primer modelo, el sistema bajo estudio es una sonda de análisis de tráfico de red y el parámetro de rendimiento más significativo es el throughput de análisis (el caudal de tráfico del sistema completo). En el segundo modelo, se estudia el subsistema de red de la sonda de análisis anterior y los parámetros de rendimiento sobre los que se incide son varios: throughput de captura, probabilidad de pérdida de paquetes, factor de utilización de CPU, frecuencia de *softirqs*, etc. Aprovechando el conocimiento adquirido en el desarrollo de los dos modelos anteriores, se plantea un tercer modelo, también basado en teoría de colas, sobre un campo de aplicación diferente: la provisión de servicios sobre una red B5G. Esta se caracteriza por estar basada en funciones virtualizadas de red denominadas VNFs que pueden estar distribuidas en diferentes nodos de la red. Tomando como base el

consumo computacional de estas entidades software, se plantea un modelo de colas para el servicio de Misión Crítica en redes 5G. Este tercer modelo tiene como parámetro de rendimiento clave la latencia de los servicios de Misión Crítica para voz y para vídeo. El modelo determina cómo debe ser el despliegue de funciones VNFs minimizando la latencia de los servicios MCX. En los tres casos, el objetivo es modelar el sistema bajo estudio siguiendo una metodología correcta. Eso supone: 1) identificar los aspectos clave del sistema a modelar; 2) proponer un modelo que represente el sistema de tal forma que asuma simplificaciones, para que su resolución sea breve, pero que a la vez recoja todos los aspectos clave identificados previamente; 3) resolver el modelo; y 4) evaluar y validar el modelo.

A continuación, se muestra con mayor detalle el trabajo desarrollado en cada una de las tres propuestas de modelado.

6.1.1. Propuesta de modelo de optimización de throughput basado en MDP

El primer modelo resuelve un problema identificado en una sonda de análisis de tráfico prototipo desarrollada por el grupo NQaS. A partir de unas medidas experimentales tomadas sobre el prototipo, se detectó que el caudal de tráfico total analizado (o throughput de análisis) de la sonda era menor del que se esperaba para la tasa de tráfico de datos inyectada en la red. Se estaba produciendo un bloqueo en la captura de los datos que indicaban una pérdida de rendimiento. Este efecto era de interés para su estudio. Para ello se ha propuesto un modelo matemático basado en teoría de colas.

En el sistema bajo estudio se distinguen dos tareas o etapas en serie, una primera de captura y una segunda de análisis. Dado que el caso observado es de recurso limitado, solamente se dispone de un procesador que debe ejecutar las dos tareas. Para representar el sistema, se propone un modelo que consiste en una cola tándem, de dos etapas, con un único servidor activo. Esto quiere decir que se tiene un único procesador que atiende a las dos tareas, moviéndose de una a otra. En cada instante solamente una de las dos etapas está atendida, es la etapa activa. Por otro lado, tanto la cola que representa a la etapa de captura como la cola asociada a la etapa de análisis son de tamaño finito.

En este punto, se plantea que haya un planificador del sistema que tome la decisión sobre cuál de las dos etapas debe estar activa en cada momento. Su objetivo es que el throughput del sistema sea lo más alto posible. Para ello, el primer paso es pasar de un modelo inicial en tiempo continuo a un modelo equivalente de tiempo discreto. Eso se consigue utilizando técnicas de uniformización. A continuación, se plantea un proceso MDP tridimensional de tiempo discreto. Las variables del proceso son el número de paquetes en la cola de captura, el número de paquetes en la cola de análisis y el estado del procesador; es decir, cuál de las dos colas está siendo atendida en ese momento. En este proceso MDP, al comienzo de cada intervalo en el que se divide el tiempo, se debe tomar una decisión sobre la acción a tomar por el planificador. Las acciones posibles son dos:

(a) activar la etapa de captura o (b) activar la de análisis. Esa decisión se toma a través de una función objetivo que hay que maximizar. Esta será una función de recompensa que premia por cada paquete analizado que sale del sistema y penaliza por cada cambio de contexto que se produce. Se entiende por cambio de contexto al cambio de estado del procesador; esto es, que estando la etapa de captura activa se pase a la de análisis o viceversa. La política óptima que el proceso MDP da como resultado es obtenida aplicando un método numérico; concretamente, el algoritmo de iteración por valor. Esta política óptima se analiza para un amplio rango de escenarios y, por simulación, se calcula el throughput de análisis que se alcanza con ella (Figura 3.8). Se observa que el throughput de la política óptima mejora de forma considerable el que se tenía en la situación inicial (Figura 3.2 con las medidas experimentales que motivaron el inicio del estudio de este modelo).

Como siguiente paso, se ha considerado la implementación de la política óptima, o una próxima a ella, en la sonda real. Se han valorado diferentes alternativas y se ha concluido que una política basada en un valor umbral es la más apropiada.

Para completar la propuesta, se ha representado el modelo mediante una red de Petri estocástica generalizada. Esta proporciona una representación gráfica que facilita reproducir la dinámica del sistema. Además, utilizando un software de simulación especializado en redes de Petri, se ha evaluado el throughput del sistema para diferentes casos y se ha hallado su máximo. Los resultados obtenidos han sido coherentes con los del modelo basado en MDP.

6.1.2. Propuesta de modelo de cola con *vacations* para sistema de captura de paquetes

El segundo modelo analiza en profundidad el sistema de captura de paquetes de Linux (sistema operativo muy común en las sondas de análisis de tráfico basadas en hardware de propósito general). El interés de esto nace también de las observaciones previas realizadas sobre el prototipo de sonda del grupo de investigación. Hay que hacer hincapié en que el estudio de este capítulo se centra en la primera etapa de procesamiento de la sonda (aquella en la que se capturan paquetes) y deja al margen la segunda etapa (aquella donde se realiza un análisis o procesado posterior de los paquetes capturados).

Antes de describir el modelo, se describe el mecanismo de captura de paquetes de Linux, detallando la trayectoria que sigue un paquete desde que entra por la tarjeta de red hasta que llega al nivel de usuario. Analizando ese procedimiento se identifican los elementos que contiene el modelo propuesto: (a) el búfer circular al que llegan los paquetes por transferencia DMA, (b) la rutina de atención de la interrupción hardware o *hardirq*, (c) el procedimiento de planificación y ejecución de la interrupción software o *softirq*. También se incorpora al modelo el funcionamiento de la *softirq* que tiene establecidos unos límites en cuanto al número de paquetes que

puede tratar en una *softirq* (a través del parámetro *budget*) y en cuanto al tiempo máximo que puede durar su ejecución.

A continuación, considerando como caso de interés el recurso limitado con una única tarjeta de red y un procesador, se propone un modelo de una cola de tamaño finito con un servidor y con *vacations*. Este modelo considera, por un lado, el proceso de captura y, por otro. el denominado tiempo de *vacation*. Se entiende como *vacation* al tiempo en el que el sistema no está activo con la tarea principal (en este caso, la captura de paquetes) y puede ocuparse de otras secundarias. Se define un diagrama bidimensional de estados del sistema donde se distinguen tres grupos: i) estados donde hay procesamiento asociado a hardirq; ii) estados con procesamiento *hardirq*; iii) estados con *vacation*. También se especifican las transiciones entre estados que tienen en cuenta las condiciones y particularidades del procedimiento. A partir del diagrama genérico mencionado, se pasa a resolver tres casos particulares a los que se les llama M1, M2 y M3.

M1 es el caso con distribuciones de tiempo exponenciales y valor infinito para el parámetro *budget* de *softirq*. Su resolución se plantea con una cadena de Markov donde no hay límite para el número de paquetes tratados dentro de una *softirq*. A partir de la cadena de Markov, se plantean las ecuaciones de balance y se calculan las probabilidades de estado en régimen estacionario haciendo un desarrollo matricial. A partir de las probabilidades de estado, se calculan los parámetros de rendimiento como: throughput de captura, probabilidad de pérdida de paquetes, disponibilidad de CPU, frecuencia de *softirq* y número medio de paquetes tratados por *softirq*.

M2 es el caso con distribuciones de tiempo exponenciales y valor finito de *budget*. Su resolución también se plantea con una cadena de Markov donde sí hay límite para el número de paquetes tratados dentro de una *softirq*. Se realiza un desarrollo similar al de M1, aunque con más estados, debido a la aparición del parámetro *budget* B, por lo que su resolución resulta más laboriosa. De este modelo también se obtienen métricas como: throughput de captura, probabilidad de bloqueo, disponibilidad de CPU, frecuencia de *softirq* y número medio de paquetes tratados por *softirq*.

M3 es el caso con distribuciones de tiempo de tipo General y valor finito para el parámetro *budget*. Es un modelo M/G/1/N con procesamiento *hardirq*, procesamiento *softirq* con disciplina de servicio limitada por *budget* y *vacations*. Para su resolución se realizan los siguientes pasos: se plantea la cadena de Markov embebida, se resuelven sus probabilidades de estado con un desarrollo matricial tras haber utilizado las transformadas de Laplace-Stieltjes de las distribuciones de tiempo de *hardirq*, *softirq* y *vacations*. A continuación, para obtener la distribución general de probabilidades del sistema, se usa la técnica de la variable suplementaria. Trabajando en el dominio de la transformada de Laplace-Stieltjes y con el desarrollo algebraico correspondiente, se llega a las expresiones de las probabilidades de estado del sistema. A partir de estas, al igual que en M1 y M2, ya es posible calcular los parámetros de rendimiento de interés.

Finalmente se procede a la evaluación de los modelos y los resultados teóricos se comparan con los obtenidos en la sonda real.

6.1.3. Propuesta de modelo de red abierta de colas para el despliegue de funciones VNF de servicios de Misión Crítica sobre red B5G

El tercer modelo plantea un problema de despliegue de funciones VNF en un escenario de servicios de Misión Crítica (MC) sobre una red B5G. En primer lugar, se describe la arquitectura que se pretende modelar y se identifican los elementos que formarán parte del modelo:

- Servidores, nodos de red, usuarios y grupos de usuarios.
- Tipos de tráfico que se diferenciarán (señalización, *floor-control*, MC-PTT, datos y vídeo).
- Conjunto de funciones VNF que se deben desplegar en la arquitectura.
- Cadenas de servicio (esto es, secuencias de funciones VNF a ejecutar) para cada una de las clases de tráfico.

Para realizar el despliegue de las funciones VNF, se selecciona la latencia de cada una de las clases de tráfico como criterio de optimización. Esto permite establecer una relación directa con los indicadores KPI que deben cumplir los servicios de Misión Crítica. Para calcular esa latencia por cada clase de tráfico, se propone un modelo basado en una red abierta de colas M/M/1. Se desarrolla la formulación teniendo en cuenta las cadenas de servicio y se llega a las expresiones de retardo medio de cada clase.

A continuación, se desarrolla el problema de optimización consistente en minimizar las latencias de servicio (con su posible indicador KPI), sujeto a una serie de restricciones de carácter computacional, energético y de estabilidad que deben cumplir los nodos donde se pueden desplegar las VNFs.

El siguiente paso es la resolución del problema. Para ello, primeramente se reformula el problema para que sea convexo; luego, se sigue un método iterativo que utiliza conjuntamente un *solver* y la heurística denominada MaxZ, para llegar al resultado final, esto es, la distribución de VNFs sobre los nodos propuesta como óptima.

Para demostrar la utilidad del modelo, se resuelven numéricamente varios ejemplos donde se puede apreciar los distintos escenarios que se pueden dar y los resultados obtenidos.

6.2. Aportaciones

El trabajo de tesis se centra en el modelado matemático orientado al análisis de rendimiento. En este sentido, se han propuesto tres modelos

diferentes, aunque todos ellos toman como base los consumos computacionales de funciones software. En cada uno de ellos, se pueden identificar una serie de aportaciones que se detallan a continuación.

6.2.1. Modelo MDP de optimización de throughput

En primer lugar cabe destacar que, hasta donde el autor de esta tesis sabe, los estudios previos de modelado sobre sistemas de monitorización de tráfico no contemplan la relación entre las etapas de captura y análisis tal y como se ha formulado en este trabajo. Generalmente los trabajos previos se centran en modelar el proceso de captura entendiendo que esa función es la que limita la capacidad de una sonda para procesar tráfico. Sin embargo, en esta tesis, los modelos desarrollados demuestran que hay una incidencia muy importante de la carga de análisis en la propia captura. También ayudan a planificar cómo debe ser el proceso de captura considerando las limitaciones de la sonda impuestas por la falta de recursos para atender la carga de análisis asociada. Por tanto, puede considerarse como aportación el hecho de tratarse de un modelo analítico novedoso basado en conceptos clásicos de procesos MDP, que tiene en cuenta las limitaciones de capacidad de procesamiento y de búferes finitos, así como la optimización del rendimiento del sistema maximizando su throughput de análisis.

A lo dicho anteriormente, se pueden añadir las siguientes aportaciones puntuales del modelo de optimización de throughput:

- **Proceso MDP discreto con política óptima adecuada.** Para resolver el modelo, se utiliza un proceso MDP cuya función de recompensa no estima directamente el throughput de análisis del sistema, sino que valora positivamente con una unidad cada paquete analizado que sale del sistema y negativamente los cambios de contexto. Los resultados demuestran lo acertado de la propuesta. Cabe añadir a esto la fiabilidad de la herramienta de simulación desarrollada para calcular el throughput que alcanza la política óptima.

- **Implementación de política en la sonda real.** Es de destacar que se consigue implementar en la sonda real una política no óptima, pero con resultados aceptables (incluso próximos a los máximos teóricos). Por tanto, dada la dificultad técnica para implementar la política óptima, se puede afirmar que la valoración de alternativas y su comparativa ha sido satisfactoria.

- **Simulación del modelo con red de Petri estocástica.** Se presenta un segundo modo para validar el modelo, consistente en representar el sistema mediante una red de Petri estocástica generalizada y evaluarlo mediante simulación con un software especializado en redes de Petri.

Por último, añadir que se presume viable aplicar, en otros entornos como pueden ser las redes 5G y B5G, la idea de emplear una red abierta de colas sobre la que se tiene un proceso MDP que busca una estrategia óptima para tomar decisiones adecuadas.

6.2.2. Modelo de cola con *vacations* para sistema de captura de Linux

También es novedoso el modelo de cola finita con procesamiento de *hardirq*, *softirq* y *vacations*, por el campo de aplicación que tiene en esta tesis. Asimismo, se pueden mencionar las siguientes aportaciones puntuales del modelo de optimización de throughput:

- **Análisis exhaustivo del mecanismo de captura de paquetes de Linux.** Se ha descrito de una manera detallada el proceso que sigue un paquete capturado por el subsistema de red de Linux y se han identificado las diferentes etapas, acciones y funciones que intervienen en el mismo.

- **Representación del procedimiento con hardirqs, softirqs y *vacations*.** Se quiere destacar la representación que se ha hecho del procedimiento de captura de paquetes, incluyendo los procesamientos *hardirq*, *softirq* y el concepto de *vacation*, su diagrama general de estados y transiciones. Su resolución permite valorar el impacto de los periodos de *vacation* en el rendimiento del sistema de captura.

- **Desarrollo matemático de los modelos.** Se han resuelto tres casos del modelo general realizando diferentes desarrollos y utilizando diferentes técnicas (cadena de Markov, cadena de Markov embebida, transformada de Laplace-Stieltjes, técnica de la variable suplementaria). El modelo M3 se ha resuelto tomando como referencia el trabajo de Lee [151]. El aspecto diferencial de este trabajo es el procesamiento *hardirq*. Se ha comprobado que, si los tiempos de procesamiento de *hardirq* son despreciables, las probabilidades de estado del modelo M3 coinciden con las del artículo de Lee.

6.2.3. Modelo de red abierta de colas para servicios de Misión Crítica

En los últimos años ha habido muchas propuestas sobre métodos para ubicar funciones VNF (*VNF placement*) [152] [153]. Para el modelo de red abierta de colas presentado en el Capítulo 5 de esta tesis, se han tomado dos como referencia: [112] y [113]. La primera por tratar el tema mediante teoría de colas; la segunda por su entorno de aplicación. Además, también se ha valorado la propuesta tecnológica sobre servicios de Misión Crítica hecha en [116]. Las aportaciones del modelo propuesto sobre la arquitectura de servicios de Misión Crítica son las siguientes:

- **Función objetivo orientada al cumplimiento de los indicadores KPI de servicios de Misión Crítica.** Al igual que en este trabajo, en [113], el escenario de aplicación son servicios de Misión Crítica (concretamente MCPTT) y se plantea un problema de optimización cuya función objetivo tiene como parámetro clave el consumo energético que se produce en los nodos donde se despliegan las VNFs. Las capacidades computacionales sí son consideradas dentro

de las restricciones impuestas y los retardos involucrados con el procesamiento de las funciones son de carácter constante. Sin embargo, en el modelado de esta tesis, la función objetivo busca minimizar los retardos de servicio, parámetro que tiene relación directa con los indicadores KPI a cumplir por los servicios de Misión Crítica. Además, al formular el modelo mediante teoría de colas, los tiempos de procesado de las VNFs no se consideran constantes, sino que son variables aleatorias más estocásticas y se ven afectados por la mayor o menor cantidad de tráfico que reciba cada VNF.

- **Clases de tráfico.** El modelo propuesto contempla el caso de múltiples tipos de tráfico. Además del tráfico MCPTT ya contemplado en [116], se incorporan todos los tipos de tráfico identificados en la arquitectura de Misión Crítica extendida (MCX): señalización, floor control, audio, datos y vídeo.

- **Estimación de tiempos de procesado de VNFs.** Tras plantear el modelo con *vacations* del Capítulo 4 se extrae la conclusión de que es importante considerar los tiempos de *vacation* que pueden tener los procesadores; esto es, tiempos en los que el procesador dispone de capacidad computacional pero no puede emplearla en una tarea determinada ya que debe atender a otra o debe esperar a que ocurra algún evento para seguir con el procedimiento establecido. Por ello, a la hora de estimar el tiempo de procesado de una función VNF, se deben tener en cuenta esos tiempos de "silencio"que también deben ser contabilizados. Es por ese motivo que, en el modelo de este trabajo, se diferencia el tiempo de procesado de VNF (t_{vnf}) y el tiempo de ocupación de CPU (t_{cpu}) asociado a esa CPU, siendo $t_{\text{cpu}<t_{\text{vnf}}}$. Esto no se contempla en ninguno de los artículos tomados como referencia.

- **Resultados relacionados con la localización de usuarios.** Una de las interpretaciones más interesantes de los resultados del modelo es ver cómo afecta el conocimiento de la localización de usuarios (es decir, saber a qué nodo de acceso están conectados los usuarios). Esta conclusión, que ya se adelantaba en el estudio realizado en [100] con técnicas de simulación, es confirmada por el modelo teórico.

6.3. Líneas futuras

Este trabajo de tesis propone varios modelos matemáticos que ayudan a predecir el rendimiento de sondas de análisis de tráfico conectadas a redes Gigabit y multi-Gigabit Ethernet. Sin embargo, la evolución de las redes y las tecnologías, la llegada de la red 5G, la provisión de servicios en la nube o el Internet de las cosas (*Internet of Things*), han traído consigo cambios muy significativos en las metodologías requeridas para una gestión de red adecuada. En la era actual, con millones de dispositivos conectándose a la nube a través de Internet, se hacen necesarias metodologías que permitan evaluar las prestaciones de los servicios de forma automática, dinámica

e inteligente para la mejora continua en la provisión de los mismos. Estas metodologías deben contemplar la asignación dinámica inteligente de recursos, la automatización inmediata de las funciones de red, las adaptaciones a las condiciones de mercado y otros muchos requerimientos que resultan imprescindibles en el marco actual de las redes y servicios de telecomunicación. Las plataformas para la virtualización, como SDN (Software Defined Networks) o NFV (Network Functions Virtualization), cada vez más extendidas, juegan también un papel muy importante en la definición de estas metodologías, dado que permiten el diseño inteligente, la configuración y parametrización dinámica y el despliegue de recursos de forma automatizada y adaptativa, de acuerdo a los requerimientos de los usuarios o las necesidades del negocio del proveedor. En este nuevo escenario, la monitorización y el análisis de tráfico de red seguirán siendo operaciones críticas y las sondas de análisis continuarán existiendo, pero estas últimas deberán adaptarse a los cambios tecnológicos. De hecho, como ya se sacó a la luz en el capítulo del estado del arte, ya existen propuestas de sondas virtuales de tráfico de red (*virtual network probes*) para entornos 5G. El modelado matemático de estos nuevos sistemas puede ser válido para predecir su rendimiento y facilitar mejoras de diseño. Por tanto, en este sentido, se abren nuevas líneas, tanto para adaptar los modelos propuestos a los nuevos escenarios de red, como para plantear novedosas ideas de modelado basándose en el conocimiento adquirido durante el desarrollo de este trabajo.

A continuación, se presentan algunas de estas líneas de investigación que se consideran interesantes para dar continuidad al trabajo de modelado que se ha realizado en esta tesis.

Captura y procesado de datos para servicios virtualizados sobre redes B5G y 6G. Dentro del plan de trabajo del grupo NQaS está el desarrollo de nuevas sondas de análisis de tráfico para las redes de futuro B5G y 6G. Para ello, se pretende utilizar arquitecturas de computación muy flexibles sobre las que se desplieguen las funciones VNF que proporcionan el servicio virtualizado de análisis de tráfico. El modelado matemático será un complemento más en el diseño de estas sondas integradas dentro de la filosofía de las nueva redes del futuro. Por ello, se pretende avanzar en la creación de modelos matemáticos que permitan estudiar de forma teórica los rendimientos de estas arquitecturas considerando los avances en virtualización de servicios, despliegue de los mismos e inteligencia artificial.

Por tanto, será un trabajo donde se combine el estudio teórico con tareas prácticas de laboratorio. La interacción entre ambos es importante. Los modelos teóricos pueden ser alimentados por datos experimentales tomados sobre los prototipos de laboratorio. Los nuevos desarrollos de software pueden ser ajustados en función de resultados obtenidos previamente de los modelos. Los resultados deberán ser validados, tanto los teóricos como los de laboratorio. Para ello, se utilizarán plataformas de simulación y emulación que están disponibles en NQaS. También se quiere avanzar en el desarrollo de modelos de simulación de redes B5G que puedan servir

de base para obtener resultados contrastables. Para ello se quiere aportar valor en la simulación de estos elementos sobre el simulador OMNET++ y otras plataformas de simulación y emulación.

Despliegue de servicios en redes B5G y 6G en función de las necesidades computacionales. En esta línea futura, se espera aportar modelado analítico para estudiar cómo se pueden atender flujos de datos en una arquitectura más distribuida de servicios virtualizados considerando que en las nuevas redes hay capacidad de orquestar y decidir dónde alojar los servicios. Las entidades NVF que permiten virtualizar servicios, incluso hardware, y que pueden ser gestionables y orquestadas conjuntamente separando sus funciones en VNFs especializadas en procesamientos específicos, deben ser gestionadas de forma inteligente. La línea de investigación del grupo NQaS sobre despliegue y orquestación inteligente de servicios en entornos 5G, B5G y 6G ya tiene dos campos de aplicación definidos: por un lado, dentro de la seguridad pública, los servicios de misión crítica, para los que en el Capítulo 5 de esta tesis ya se ha propuesto un primer modelo basado en una red de colas; por otro lado, la fabricación avanzada. Esta última actividad se enfoca hacia el despliegue de servicios sobre redes privadas 5G orientados a la digitalización de los procesos y la convergencia IT/OT (tecnología de la información/tecnología operativa) para la mejora de los ciclos productivos. Se basa en el diseño específico del modelo de datos para la monitorización de información a distintos niveles (IoT, ERP, monitorización de red) para una gestión automatizada que habilite el mantenimiento preventivo, la fabricación con cero defectos y un control de la gobernanza de los datos en la cadena de suministro extendida. Los modelos matemáticos que se propongan deberán recoger las características específicas de cada uno de esos campos de aplicación

Análisis integral de la calidad de servicio (QoX). El grupo NQaS cuenta con otra línea de investigación denominada "Análisis integral de la calidad de servicio (QoX)" que está relacionada con la calidad de servicio. Sus investigaciones iniciales estaban centradas sobre todo en las métricas y la vertiente más objetiva de la calidad de servicio. Ahora, ha evolucionado hacia un modelo más global para la gestión integral de la QoS (QoX), denominado QoXfera [154], siendo X la experiencia del usuario con el servicio (QoE, Quality of user Experience) o su satisfacción con el mismo, la calidad de negocio de los proveedores y operadores (QoBiz, Quality of Business), la confianza de los datos capturados (QoD, Quality of Data), el alcance y trascendencia de la información capturada y procesada para poder tomar decisiones (QoI, Quality of Information) y otras muchas vertientes de la calidad que tienen influencia actualmente en la prestación de los servicios. En este contexto, es necesario disponer de un soporte de técnicas eficientes para la captura y procesado de datos. Su análisis y modelado matemático podría ser una línea de continuidad de esta tesis.

6.4. Conclusiones finales

Como conclusiones finales de este trabajo de tesis, puede afirmarse que la construcción de modelos matemáticos que permiten estudiar el rendimiento teórico de sistemas de captura y análisis de tráfico de redes resulta de interés en el proceso de diseño de ese tipo de dispositivos. La posibilidad de predecir comportamientos y aspectos cuantitativos de rendimiento es lo que hace atractivo al modelado. En el caso de las sondas de tráfico de red, con los modelos propuestos, se puede estimar la efectividad del sistema bajo diferentes condiciones de tráfico de red (intensidad del flujo de datos que entra a la sonda desde la red) y de carga de análisis (intensidad del procesamiento que requiere el flujo de datos en el sistema de análisis). Si los operadores que hacen seguimiento de un sistema de análisis de tráfico o los responsables de su diseño detectan alguna clave que impacta negativamente sobre el rendimiento, su análisis y modelado puede dar una respuesta rápida y válida para, después, realizar el ajuste que subsane la pérdida de rendimiento.

Además, la base del modelado utilizado para las sondas también es "exportable" a otro tipo de entornos, tal y como se ha visto al aplicar un modelo de colas sobre un servicio virtualizado de Misión Crítica sobre red 5G. La característica común que ha hecho esto posible, ha sido el análisis de los consumos computacionales de las funciones software que forman parte de la provisión del servicio extremo a extremo. A partir de ahí, se han construido modelos diferentes (cola tándem con servidor activo único, cola con *vacations*, red abierta de colas), cada uno adaptado al problema o sistema específico que es objeto de estudio. El siguiente paso ha sido resolver matemáticamente los modelos; cada uno con técnicas diferentes: proceso MDP, cadenas de Markov, cadenas de Markov embebidas, técnica de la variable suplementaria, transformada de Laplace-Stieltjes, convexización del problema de optimización sujeto a una serie de restricciones... Posteriormente, se ha procedido con la validación de los modelos, evaluando numéricamente los modelos con valores significativos y contrastando los resultados teóricos o bien con los de un sistema real o bien con los extraídos de una simulación.

Asimismo, se puede decir que se han cumplido los objetivos marcados:

- Se han generado tres modelos matemáticos que responden a los efectos de interés analizados en cada uno de los casos. Se han seguido las pautas de una metodología válida: realizando simplificaciones de los sistemas reales a modelar pero, a la vez, recogiendo todos sus elementos clave; siendo rigurosos en los desarrollos matemáticos empleados para resolver los modelos; validando los modelos con fiabilidad.

- Se ha alcanzado el objetivo específico relacionado con el primer modelo al obtener una estrategia de planificación teórica/óptima que resuelve el problema de pérdida de throughput e, igualmente, se ha propuesto una estrategia implementable (no óptima) cuyos resultados mejoran significativamente la situación inicial.

- En lo que respecta al segundo modelo, se ha logrado analizar exhaustivamente el comportamiento del sistema de captura y se ha podido respresentar de una forma adecuada mediante el sistema de colas con *vacations*; este permite calcular el throughput de captura y otras métricas de rendimiento, analizando el efecto que tiene sobre ellos la variación de los tiempos de *vacation* o del parámetro *budget*.

- Finalmente, el tercer modelo ha confirmado que la base de las técnicas de modelado utilizadas en los dos modelos anteriores es aplicable en el servicio virtualizado de Misión Crítica sobre red 5G. Asimismo, se ha propuesto un modelo cuya utilidad ha sido confirmada sobre un ejemplo teórico de una arquitectura simplificada. Puede ser el primer paso para el diseño de una nueva forma de implementar servicios de Misión Crítica. No obstante, indicar que esto último requiere un estudio mucho más profundo, por ejemplo, el relativo al cumplimiento de estándares necesarios para llevar a cabo un despliegue de VNF en ese entorno de sistemas de Misión Crítica.

Cabe señalar que el trabajo realizado tiene una clara continuación apoyando las otras líneas de investigación del grupo, tal y como se ha mencionado en el apartado de líneas futuras. Y también son de destacar las contribuciones que se han realizado en esta tesis, útiles para la divulgación científica de este trabajo.

6.5. Contribuciones

Como resultado de la investigación de este trabajo de tesis, ha habido un conjunto de contribuciones que se recogen en este apartado. Primeramente se listarán las publicaciones en revistas y, en segundo lugar, los artículos presentados en congresos.

6.5.1. Publicaciones

A continuación, se detallan las publicaciones en revistas relacionadas con los contenidos presentados en esta tesis. Se aporta información sobre el *Journal Citation Reports (JCR) Impact Factor (IF)* de las revistas.

- **Publicaciones en revistas**

  [**R1**] L. Zabala, J. Doncel, A. Ferro, "Modeling a Linux Packet-Capturing System with a Queueing System with Vacations", *Mathematics*, vol. 11, no. 7:1663, Marzo 2023. DOI: 10.3390/math11071663
  JCR (año 2021) IF 2.592: Rank 21/333 **Q1** *Mathematics*.

  [**R2**] L. Zabala, J. Doncel, A. Ferro, "Optimality of a Network Monitoring Agent and Validation in a Real Probe", *Mathematics*, vol. 11, no. 3:610, Enero 2023. DOI: 10.3390/math11030610
  JCR (año 2021) IF 2.592: 21/333 **Q1** *Mathematics*.

[**R3**] R. Solozabal, J. Ceberio, A. Sanchoyerto, L. Zabala, B. Blanco, F. Liberal, "Virtual Network Function Placement Optimization with Deep Reinforcement Learning", *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 2, págs: 292–303, Febrero 2020. DOI: 10.1109/JSAC.2019.2959183
JCR (año 2021) IF 13.081: 3/93 **Q1** *Telecommunications*; 8/276 **Q1** *Engineering, Electrical & Electronic.*

[**R4**] E. Ibarrola, E. Saiz, L. Zabala, L. Cristobo, J. Xiao, "A New Global Quality of Service Model: QoXphere", *IEEE Communications Magazine*, vol. 52, no. 1, págs: 193–199, Enero 2014. DOI: 10.1109/MCOM.2014.6710083
JCR (año 2021) IF 9.030: 7/93 **Q1** *Telecommunications*; 22/276 **Q1** *Engineering, Electrical & Electronic.*

[**R5**] L. Zabala, A. Pineda, A. Ferro, L. Alonso, "A Kernel-Level Traffic Probe to Capture and Analyze Data Flows with Priorities", *International Journal on Recent Trends in Engineering & Technology*, vol. 8, no. 1, págs: 47–50, Enero 2013.

- **Capítulos de libro**

[**CL1**] L. Zabala, A. Ferro, A. Pineda, A. Muñoz, "Modelling a Network Traffic Probe Over a Multiprocessor Architecture". En: Jesús Hamilton, editor. *Telecommunications Networks - Current Status and Future Trends*, págs: 303–328. InTech; Marzo 2012.

6.5.2. Artículos y ponencias en congresos

A continuación, se detallan las contribuciones a congresos relacionadas con los contenidos de esta tesis. En primer lugar, se listarán las de congresos internacionales y, luego, las de congresos nacionales.

- **Contribuciones a congresos internacionales**

[**C1**] L. Zabala, R. Solozabal, A. Ferro, B. Blanco, "Model of a Virtual Firewall based on Stochastic Petri Nets", en *2018 IEEE 17th International Symposium on Network Computing and Applications (NCA)*, Cambridge (EEUU), Noviembre 2018, págs. 311–314.

[**C2**] L. Zabala, R. Solozabal, A. Ferro, B. Blanco, "Performance Analysis of a Network Sensor's Packet Processing System using Generalized Stochastic Petri Nets", en *15th ACM International Symposium on Performance Evaluation of Wireless Ad Hoc, Sensor, & Ubiquitous Networks*, Montreal (Canadá), Octubre 2018, págs. 63–70.

[**C3**] L. Zabala, A. Pineda, A. Ferro, D. Fernández, "Comparing Network Traffic Probes based on Commodity Hardware", en *ICN 2014: The Thirteenth International Conference on Networks*, Niza (Francia), Febrero 2014, págs. 261–267.

[**C4**] E. Ibarrola, E. Saiz, J. Xiao, L. Zabala, L. Cristobo, "QoXphere: A New QoS Framework for Future Networks", en *2013 Proceedings of ITU Kaleidoscope: Building Sustainable Communities*, Kyoto (Japón), Abril 2013, págs. 119–126.

[**C5**] A. Pineda, L. Zabala, A. Ferro, "Network architecture to automatically test traffic monitoring systems", en *Proceedings of the Mosharaka International Conference on Communications and Signal Processing (MIC-CSP2012)*, Barcelona (España), Abril 2012.

[**C6**] L. Zabala, A. Ferro, A. Pineda, "Modeling Packet Processing Time in a Multiprocessor Network Traffic Monitoring System", en *The 2012 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA)*, Las Vegas (Estados Unidos), Julio 2012.

[**C7**] L. Zabala, A. Ferro, A. Pineda, "Modelling Packet Capturing in a Traffic Monitoring System based on Linux", en *The International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS 2012)*, Génova (Italia), Julio 2012.

[**C8**] L. Zabala, A. Ferro, C. Perfecto, E. Ibarrola, J.L. Jodra, "LabQoS: A platform for network test environments", en *Proceedings of the 2011 ITU Kaleidoscope Academic Conference: The fully networked human?*, Ciudad del Cabo (Sudáfrica), Diciembre 2011, págs. 189–198.

- **Contribuciones a congresos nacionales**

[**C9**] L. Cristobo, L. Zabala, E. Ibarrola, A. Ferro, F. Liberal, "Metodología para la gestión de la QoX basada en el aprendizaje automático", en *XIV Jornadas de Ingeniería Telemática (JITEL 2019)*, Zaragoza (España), 22-24 octubre 2019, págs. 138–142.

[**C10**] L. Zabala, A. Ferro, A. Nieva, "Modelo de colas con vacations aplicado a un sistema de captura de paquetes", en *XIII Jornadas de Ingeniería Telemática (JITEL 2017)*, Valencia (España), 27-29 septiembre 2017, págs. 132–139.

[**C11**] L. Zabala, A. Ferro, A. Pineda, "Propuesta de modelo con redes de Petri estocásticas para optimización de una sonda de análisis de tráfico de datos", en *XI Jornadas de Ingeniería Telemática (JITEL 2013)*, Granada (España), 28-30 octubre 2013, págs. 227–234.

[**C12**] J. Domingo, A. Pineda, A. Ferro, L. Zabala, "Adaptación y extensión de una herramienta de análisis de tráfico para redes", en *XXVIII Simposio Nacional de la Unión Científica Internacional de Radio (URSI 2013)*, Santiago de Compostela (España), 12-14 septiembre 2013, comunicación nº 193.

[**C13**] I. Blanco, A. Morán, A. Ferro, L. Zabala, A. Pineda, "Arquitectura de generación e inyección de tráfico sintético configurable

en FPGA", en *XXVII Simposium Nacional de la Unión Científica Internacional de Radio (URSI 2012)*, Elche (España), 12-14 septiembre 2012.

[**C14**] D. Fernández, A. Ferro, L. Zabala, A. Pineda, "Integración de PF_RING en el sensor de análisis de tráfico Adviser", en *XXVII Simposium Nacional de la Unión Científica Internacional de Radio (URSI 2012)*, Elche (España), 12-14 septiembre 2012.

[**C15**] A. Ferro, L. Zabala, A. Pineda, I. Blanco, "Traffic Analysis in High-Speed Networks: A Research Line for Designing and Modelling Traffic Monitoring Solutions", en *Proceedings of the 2nd Workshop "Future Internet: Efficiency in high-speed networks" (W-FIERRO 2012)*, Cartagena (España), 19-20 julio 2012, págs. 89–96.

[**C16**] L. Alonso, A. Ferro, L. Zabala, A. Pineda, "Diseño para la captura y análisis de flujos con prioridad en un sensor de tráfico a nivel de kernel", en *XXVI Simposium Nacional de la Unión Científica Internacional de Radio (URSI 2011)*, Leganés (España), 7-9 septiembre 2011.

[**C17**] L. Zabala, A. Ferro, A. Pineda, "Modeling a Multiprocessor Traffic Capturing and Analysis System", en *Proceedings of the 1st Workshop "Future Internet: Efficiency in high-speed networks" (W-FIERRO 2011)*, Cartagena (España), 7-8 julio 2011, comunicación nº 14.

[**C18**] A. Pineda, A. Ferro, A. Muñoz, L. Zabala, "Caracterización Temporal de GNU/Linux para el Diseño de un Inyector Software de Tráfico Sintético", en *XXV Simposium Nacional de la Unión Científica Internacional de Radio (URSI 2010)*, Bilbao (España), 15-17 septiembre 2010.

[**C19**] A. Pineda, L. Zabala, A. Ferro, A. Muñoz, "Estudio de los Mecanismos de Espera de GNU/Linux para el Diseño de un Inyector Software de Tráfico Sintético", en *XXV Simposium Nacional de la Unión Científica Internacional de Radio (URSI 2010)*, Bilbao (España), 15-17 septiembre 2010.

Bibliografía

[1] M. S. Obaidat y N. A. Boudriga, *Fundamentals of performance evaluation of computer and telecommunication systems*. John Wiley & Sons, 2010.

[2] R. Jain, *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*. John Wiley & Sons, 1991.

[3] *The Mobile Economy 2022*, `https://www.gsma.com`, accessed: 2023-04-10.

[4] *2023 Global Internet Phenomena Report*, `https://www.sandvine.com/global-internet-phenomena-report-2023`, accessed: 2023-04-20.

[5] B. Li, J. Springer, G. Bebis et al. «A survey of network flow applications», *Journal of Network and Computer Applications*, vol. 36, n.º 2, págs. 567-581, 2013.

[6] N. T. Stevens y J. D. Wilson, «The past, present, and future of network monitoring: A panel discussion», *Quality Engineering*, vol. 33, n.º 4, págs. 715-718, 2021.

[7] H.-J. Liao, C.-H. R. Lin, Y.-C. Lin et al. «Intrusion detection system: A comprehensive review», *Journal of Network and Computer Applications*, vol. 36, n.º 1, págs. 16-24, 2013.

[8] H. I. Abdel-Gawad, D. Baleanu y A. H. Abdel-Gawad, «Unification of the different fractional time derivatives: An application to the epidemic-antivirus dynamical system in computer networks», *Chaos, Solitons & Fractals*, vol. 142, pág. 110 416, 2021.

[9] A. Pramanik, S. Sarkar y J. Maiti, «A real-time video surveillance system for traffic pre-events detection», *Accident Analysis & Prevention*, vol. 154, pág. 106 019, 2021.

[10] S. Lee, K. Levanti y H. S. Kim, «Network monitoring: Present and future», *Computer Networks*, vol. 65, págs. 84-98, 2014.

[11] *Endace Ltd*, `https://www.endace.com`, accessed: 2023-01-11.

[12] J. C. Mogul y K. K. Ramakrishnan, «Eliminating receive livelock in an interrupt-driven kernel», *ACM Transactions on Computer Systems*, vol. 15, n.º 3, págs. 217-252, 1997.

[13] B. Vanderpool y Z. Smith, «A Linux implementation of HIP», *Project report, University of Wisconsin, MD, http://web. demigod. org/˜ zak/documents/college/ece750-report/pdf*, 1998.

[14] L. Rizzo, «Device polling support for FreeBSD», en *BSDConEurope Conference*, vol. 31, 2001.

[15] R. Prasad, M. Jain y C. Dovrolis, «Effects of interrupt coalescence on network measurements», en *Passive and Active Network Measurement: 5th International Workshop, PAM 2004, Antibes Juan-les-Pins, France, April 19-20, 2004. Proceedings 5*, Springer, 2004, págs. 247-256.

[16] L. Deri, «High-speed dynamic packet filtering», *Journal of Network and Systems Management*, vol. 15, n.º 3, págs. 401-415, 2007.

[17] S. McCanne y V. Jacobson, «The BSD Packet Filter: A New Architecture for User-level Packet Capture», en *USENIX winter*, vol. 46, 1993.

[18] H. Bos, W. De Bruijn, M.-L. Cristea et al. «FFPF: Fairly Fast Packet Filters.», en *OSDI*, vol. 4, 2004, págs. 24-24.

[19] Z. Wu, M. Xie y H. Wang, «Swift: A Fast Dynamic Packet Filter.», en *NSDI*, vol. 8, 2008, págs. 279-292.

[20] M. A. Vieira, M. S. Castanho, R. D. Pacífico et al. «Fast packet processing with ebpf and xdp: Concepts, code, challenges, and applications», *ACM Computing Surveys (CSUR)*, vol. 53, n.º 1, págs. 1-36, 2021.

[21] C.-L. Hung y S.-W. Guo, «Fast parallel network packet filter system based on CUDA», *International Journal of Networked and Distributed Computing*, vol. 2, n.º 4, págs. 198-210, 2014.

[22] *Official website of tcpdump and libpcap*, `http://www.tcpdump.org`, accessed: 2023-04-10.

[23] *Wireshark*, `https://www.wireshark.com`, accessed: 2023-04-23.

[24] W. E. Leland y D. V. Wilson, «High Time-Resolution Measurement and Analysis of LAN Traffic: Implications for LAN Interconnection», en *Proceedings IEEE INFOCOM '91, The Conference on Computer Communications, Tenth Annual Joint Conference of the IEEE Computer and Communications Societies, Networking in the 90s, Bal Harbour, Florida, USA, April 7-11, 1991*, IEEE Computer Society, 1991, págs. 1360-1366. DOI: `10.1109/INFCOM.1991.147663`.

[25] *IEEE Standards Association*, `https://standards.ieee.org`, accessed: 2023-04-19.

[26] *Ethernet Alliance*, `https://www.ethernetalliance.org`, accessed: 2023-04-19.

[27] F. Schneider y J. Wallerich, «Performance evaluation of packet capturing systems for high-speed networks», en *Proceedings of the 2005 ACM conference on Emerging network experiment and technology*, 2005, págs. 284-285.

[28] P. Wang y Z. Liu, «Operating System Support for High-Performance Networking, A Survey», *Journal of China Universities of Posts and Telecommunications*, págs. 32-42, 2004.

[29] R. Giladi, *Network processors: architecture, programming, and implementation*. Morgan Kaufmann, 2008.

[30] T. Wolf, R. Ramaswamy, S. Bunga et al. «An architecture for distributed real-time passive network measurement», en *14th IEEE International Symposium on Modeling, Analysis, and Simulation*, IEEE, 2006, págs. 335-344.

[31] G. Antichi, S. Giordano, D. J. Miller et al. «Enabling open-source high speed network monitoring on NetFPGA», en *2012 IEEE Network Operations and Management Symposium*, IEEE, 2012, págs. 1029-1035.

[32] L. Deri y N. SpA, «nProbe: an open source netflow probe for gigabit networks», en *TERENA Networking Conference*, 2003, págs. 1-4.

[33] *Cisco IOS NetFlow*, `https://www.cisco.com`, accessed: 2023-04-19.

[34] *ntop*, `https://www.ntop.org`, accessed: 2023-02-24.

[35] L. Deri, «Improving passive packet capture: Beyond device polling», en *Proceedings of SANE*, Amsterdam, Netherlands, vol. 2004, 2004, págs. 85-93.

[36] J. Hurwitz y W.-C. Feng, «End-to-end performance of 10-gigabit Ethernet on commodity systems», *IEEE Micro*, vol. 24, n.º 1, págs. 10-22, 2004.

[37] W.-c. Feng, P. Balaji, C. Baron et al. «Performance characterization of a 10-Gigabit Ethernet TOE», en *13th Symposium on High Performance Interconnects (HOTI'05)*, IEEE, 2005, págs. 58-63.

[38] S.-H. Han, M.-S. Kim, H.-T. Ju et al. «The Architecture of NG-MON: A Passive Network Monitoring System for High-Speed IP Networks», en *Management Technologies for E-Commerce and E-Business Applications*, Springer Berlin Heidelberg, 2002, págs. 16-27.

[39] P. Arlos, M. Fiedler y A. A. Nilsson, «A distributed passive measurement infrastructure», en *Passive and Active Network Measurement: 6th International Workshop, PAM 2005, Boston, MA, USA, March 31-April 1, 2005. Proceedings 6*, Springer, 2005, págs. 215-227.

[40] A. Munoz, A. Ferro, F. Liberal et al. «A Kernel-Level Monitor over Multiprocessor Architectures for High-Performance Network Analysis with Commodity Hardware», en *2007 International Conference on Sensor Technologies and Applications (SENSORCOMM 2007)*, IEEE, 2007, págs. 457-462.

[41] L. Braun, A. Didebulidze, N. Kammenhuber et al. «Comparing and Improving Current Packet Capturing Solutions based on Commodity Hardware», en *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement, IMC'10, Melbourne, Australia, November 1-3, 2010*, ACM, 2010, págs. 206-217. DOI: `10.1145/1879141.1879168`.

[42] *Project BISmark*, `https://github.com/projectbismark`, accessed: 2023-04-19.

[43] *TRAMMS Traffic Measurements and Models in Multi-Service networks*, `https://www.celticnext.eu/project-tramms`, accessed: 2023-04-19.

[44] *COST Action IC0703-Data Traffic Monitoring and Analysis: theory, techniques, tools and applications for the future networks*, `https://www.cost.eu/actions/IC0703`, accessed: 2023-04-23.

[45] V. Moreno, J. Ramos, P. M. S. del Río et al. «Commodity packet capture engines: Tutorial, cookbook and applicability», *IEEE Communications Surveys & Tutorials*, vol. 17, n.º 3, págs. 1364-1390, 2015.

[46] *Receive Side Scaling on Intel Network adapters*, `https://www.intel.com`, accessed: 2023-04-19.

[47] A. Cardigliano, L. Deri, J. Gasparakis et al. «vPF_RING: towards wire-speed network monitoring using virtual machines», en *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*, ACM, 2011, págs. 533-548.

[48] S. Han, K. Jang, K. Park et al. «PacketShader: a GPU-accelerated software router», *ACM SIGCOMM Computer Communication Review*, vol. 41, n.º 4, págs. 195-206, 2011.

[49] L. Rizzo, «netmap: A Novel Framework for Fast Packet I/O.», en *USENIX Annual Technical Conference*, 2012, págs. 101-112.

[50] N. Bonelli, A. Di Pietro, S. Giordano et al. «On multi–gigabit packet capturing with multi–core commodity hardware», en *Passive and Active Measurement*, Springer, 2012, págs. 64-73.

[51] V. Moreno, P. M. S. Del Rio, J. Ramos et al. «Packet storage at multi-gigabit rates using off-the-shelf systems», en *2014 IEEE Intl Conf on High Performance Computing and Communications, 2014 IEEE 6th Intl Symp on Cyberspace Safety and Security, 2014 IEEE 11th Intl Conf on Embedded Software and Syst (HPCC, CSS, ICESS)*, IEEE, 2014, págs. 486-489.

[52] *DPDK Project*, `https://www.dpdk.org`, accessed: 2023-02-24.

[53] *OpenOnload*, `https://www.openonload.org`, accessed: 2023-02-24.

[54] S. Gallenmüller, P. Emmerich, F. Wohlfart et al. «Comparison of frameworks for high-performance packet IO», en *2015 ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS)*, IEEE, 2015, págs. 29-38.

[55] P.-W. Tsai, C.-W. Tsai, C.-W. Hsu et al. «Network monitoring in software-defined networking: A review», *IEEE Systems Journal*, vol. 12, n.º 4, págs. 3958-3969, 2018.

[56] C.-T. Yang, S.-T. Chen, J.-C. Liu et al. «Implementation of a real-time network traffic monitoring service with network functions virtualization», *Future Generation Computer Systems*, vol. 93, págs. 687-701, 2019.

[57] A. J. Aparcana-Tasayco, F. Mendoza-Cardenas y D. Diaz-Ataucuri, «Open and Interactive NMS for Network Monitoring in Software Defined Networks», en *2022 International Conference on Electronics, Information, and Communication (ICEIC)*, IEEE, 2022, págs. 1-4.

[58] M. S. Aktas, «Hybrid cloud computing monitoring software architecture», *Concurrency and Computation: Practice and Experience*, vol. 30, n.º 21, e4694, 2018.

[59] G. Julián-Moreno, R. Leira, J. E. L. de Vergara et al. «On the feasibility of 40 Gbps network data capture and retention with general purpose hardware», en *Proceedings of the 33rd Annual ACM Symposium on Applied Computing*, 2018, págs. 970-978.

[60] R. Leira, G. Julián-Moreno, I. González et al. «Performance assessment of 40 Gbit/s off-the-shelf network cards for virtual network probes in 5G networks», *Computer Networks*, vol. 152, págs. 133-143, 2019.

[61] A. Badshah, A. Jalal, U. Farooq et al. «Service level agreement monitoring as a service: an independent monitoring service for service level agreements in clouds», *Big Data*, 2022.

[62] *Kubernetes*, `https://kubernetes.io/`, accessed: 2023-04-19.

[63] R. Pérez, J. García-Reinoso, A. Zabala et al. «An experimental publish-subscribe monitoring assessment to Beyond 5G networks», *EURASIP Journal on Wireless Communications and Networking*, vol. 2021, n.º 1, págs. 1-27, 2021.

[64] R. Perez, J. Garcia-Reinoso, A. Zabala et al. «A Distributed Framework Based on Publish-Subscribe to Monitor Beyond 5G Networks», 2020.

[65] 3GPP TS 23.288, «Architecture enhancements for 5G System (5GS) to support network data analytics services»,

[66] J. Groenendijk y Z. Lan, «5G Management and Orchestration Architecture Framework», *Journal of ICT Standardization*, págs. 81-92, 2019.

[67] R. I. Pereira, I. M. Dupont, P. C. Carvalho et al. «IoT embedded linux system based on Raspberry Pi applied to real-time cloud monitoring of a decentralized photovoltaic plant», *Measurement*, vol. 114, págs. 286-297, 2018.

[68] E. Jo y H. Yoo, «Implementation of cloud monitoring system based on open source monitoring solution», en *Software Engineering in IoT, Big Data, Cloud and Mobile Computing*, Springer, 2021, págs. 181-190.

[69] V. B. Iversen, *Teletraffic Engineering Handbook*. ITU-D, 2006.

[70] J.-Y. Le Boudec, *Performance evaluation of computer and communication systems*. EPFL Press, 2010.

[71] M. Harchol-Balter, *Performance modeling and design of computer systems: queueing theory in action*. Cambridge University Press, 2013.

[72] G. Giambene, *Queuing Theory and Telecommunications: Networks and Applications*. Springer Nature, 2021.

[73] S. Poryazov, E. Saranova e I. Ganchev, «Conceptual and analytical models for predicting the quality of service of overall telecommunication systems», en *Autonomous Control for a Reliable Internet of Services: Methods, Models, Approaches, Techniques, Algorithms, and Tools*, Springer International Publishing Cham, 2018, págs. 151-181.

[74] V. Andonov, S. Poryazov, A. Otsetova et al. «A queue in overall telecommunication system with quality of service guarantees», en *Future Access Enablers for Ubiquitous and Intelligent Infrastructures: 4th EAI International Conference, FABULOUS 2019, Sofia, Bulgaria, March 28-29, 2019, Proceedings 283*, Springer, 2019, págs. 243-262.

[75] V. Kartashevskiy y M. Buranova, «OpenFlow-based software-defined networking queue model», en *Distributed Computer and Communication Networks: 24th International Conference, DCCN 2021, Moscow, Russia, September 20–24, 2021, Revised Selected Papers*, Springer, 2022, págs. 62-76.

[76] A. L. Aliyu y J. Diockou, «An Analytical Queuing Model Based on SDN for IoT Traffic in 5G», en *Advanced Information Networking and Applications: Proceedings of the 37th International Conference on Advanced Information Networking and Applications (AINA-2023), Volume 3*, Springer, 2023, págs. 435-445.

[77] K. Salah y K. El-Badawi, «Evaluating system performance in gigabit networks», en *28th Annual IEEE International Conference on Local Computer Networks, 2003. LCN'03. Proceedings.*, IEEE, 2003, págs. 498-505.

[78] K. Salah y K. El-Badawi, «On modelling and analysis of receive livelock and CPU utilization in high-speed networks», *International Journal of Computers and Applications*, vol. 28, n.º 2, págs. 162-169, 2006. DOI: 10.1080/1206212X.2006.11441800.

[79] K. El-Badawi y F. Haidari, «Performance analysis and comparison of interrupt-handling schemes in Gigabit networks», *Computer Communications*, vol. 30, n.º 17, págs. 3425-3441, 2007.

[80] K. Salah y M. Hamawi, «Comparative packet-forwarding measurement of three popular operating systems», *Journal of Network and Computer Applications*, vol. 32, n.º 5, págs. 1039-1048, 2009.

[81] K. Salah y A. Kahtani, «Improving snort performance under linux», *IET communications*, vol. 3, n.º 12, págs. 1883-1895, 2009.

[82] K. Salah, «Modelling and analysis of PC-based software routers», *Computer Communications*, vol. 33, n.º 12, págs. 1462-1470, 2010.

[83] K. Salah, K. Elbadawi y R. Boutaba, «Performance modeling and analysis of network firewalls», *Network and Service Management, IEEE Transactions on*, vol. 9, n.º 1, págs. 12-21, 2012.

[84] K. Salah, «A Queueing Model to Achieve Proper Elasticity for Cloud Cluster Jobs», en *Cloud Computing (CLOUD), 2013 IEEE Sixth International Conference on*, IEEE, 2013, págs. 755-761.

[85] K. Salah, K. Elbadawi y R. Boutaba, «An analytical model for estimating cloud resources of elastic services», *Journal of Network and Systems Management*, vol. 24, págs. 285-308, 2016.

[86] K. Salah y S. El Kafhali, «Performance modeling and analysis of hypoexponential network servers», *Telecommunication Systems*, págs. 1-12, 2017.

[87] W. Wu, M. Crawford y M. Bowden, «The performance analysis of Linux networking – packet receiving», *Computer Communications*, vol. 30, n.º 5, págs. 1044-1057, 2007.

[88] M. Jarschel, S. Oechsner, D. Schlosser et al. «Modeling and performance evaluation of an OpenFlow architecture», en *2011 23rd International Teletraffic Congress (ITC)*, IEEE, 2011, págs. 1-7.

[89] S. Zapechnikov, N. Miloslavskaya y A. Tolstoy, «Modeling of next-generation firewalls as queueing services», en *Proceedings of the 8th International Conference on Security of Information and Networks*, 2015, págs. 250-257.

[90] S. Xuan, W. Yang, H. Dong et al. «Performance evaluation model for application layer firewalls», *PLoS One*, vol. 11, n.º 11, e0167280, 2016.

[91] S. Xuan, D. Man, J. Zhang et al. «Mathematical performance evaluation model for mobile network firewall based on queuing», *Wireless Communications and Mobile Computing*, vol. 2018, 2018.

[92] R. Bolla, R. Bruschi, A. Carrega et al. «Green networking with packet processing engines: Modeling and optimization», *IEEE/ACM Transactions On Networking*, vol. 22, n.º 1, págs. 110-123, 2014.

[93] A. G. M. Ibrahim, M. E. Khedr y M. Shaheen, «Power Consumption of Packet Processing Engines and Interfaces of Edge Router: Measurements and Modeling», en *ICNS 2016, The Twelfth International Conference on Networking and Services*, IARIA, Lisbon, Portugal, 2016, págs. 23-28.

[94] X. Li, F. Ren y B. Yang, «Modeling and analyzing the performance of high-speed packet I/O», *Tsinghua Science and Technology*, vol. 26, n.º 4, págs. 426-439, 2021.

[95] 3GPP TS 23.501, *Technical Specification Group Services and System Aspects: 1. System Architecture for the 5G System - Stage 2*, ver. 15.0.0 Release 15: The 3rd Generation Partnership Project (3GPP), Internet-Draft, ene. de 2018.

[96] G. ETSI, «Network Functions Virtualization (NFV); Architectural Framework», *ETsI Gs NFV*, vol. 2, pág. V1, 2014.

[97] M. ETSI, «Mobile edge computing (mec); framework and reference architecture», *ETSI, DGS MEC*, vol. 3, págs. 1-18, 2016.

[98] *IPLOOK End-to-end Mobile Network Solution*, `https://www.iplook.com/products/mission-critical-push-to-talk`, accessed: 2023-05-02.

[99] A. Sanchoyerto, «Análisis del despliegue de Comunicaciones de Misión Crítica sobre redes 4G y 5G», Tesis doct., Universidad del País Vasco, 2021.

[100] A. Sanchoyerto, B. Blanco, E. Aldecoa et al. «Análisis despliegue de servicios de misión crítica en el extremo de la red», en *JITEL 2021 Libro de Actas: XV Jornadas de Ingeniería Telemática, A Coruña 2021*, Universidad de La Coruña, 2021, págs. 169-175.

[101] M. Cakmak, A. Zafer y C. Torun, «Performance comparison of queue management algorithms in lte networks using NS-3 simulator», *Tehnički vjesnik*, vol. 28, n.º 1, págs. 135-142, 2021.

[102] D. Fadhil y R. Oliveira, «Estimation of 5G Core and RAN End-to-End Delay through Gaussian Mixture Models», *Computers*, vol. 11, n.º 12, pág. 184, 2022.

[103] O. Adamuz-Hinojosa, V. Sciancalepore, P. Ameigeiras et al. «A Stochastic Network Calculus (SNC)-based model for planning B5G uRLLC RAN slices», *IEEE Transactions on Wireless Communications*, 2022.

[104] L. Chinchilla-Romero, J. Prados-Garzon, P. Ameigeiras et al. «5G infrastructure network slicing: E2E mean delay model and effectiveness assessment to reduce downtimes in industry 4.0», *Sensors*, vol. 22, n.º 1, pág. 229, 2021.

[105] Q. Ye, W. Zhuang, X. Li et al. «End-to-end delay modeling for embedded VNF chains in 5G core networks», *IEEE Internet of Things Journal*, vol. 6, n.º 1, págs. 692-704, 2018.

[106] J. Xin, Q. Zhu, G. Liang et al. «Performance analysis of cellular networks with D2D communication based on queuing theory model», *KSII Transactions on Internet and Information Systems (TIIS)*, vol. 12, n.º 6, págs. 2450-2469, 2018.

[107] S. Fowler, C. H. Häll, D. Yuan et al. «Analysis of vehicular wireless channel communication via queueing theory model», en *2014 IEEE International Conference on Communications (ICC)*, IEEE, 2014, págs. 1736-1741.

[108] M. Polese, L. Bonati, S. D'Oro et al. «Understanding O-RAN: Architecture, interfaces, algorithms, security, and research challenges», *IEEE Communications Surveys & Tutorials*, 2023.

[109] S. A. AlQahtani y W. A. Alhomiqani, «A multi-stage analysis of network slicing architecture for 5G mobile networks», *Telecommunication Systems*, vol. 73, págs. 205-221, 2020.

[110] J. Prados-Garzon, P. Ameigeiras, J. J. Ramos-Munoz et al. «Performance modeling of softwarized network services based on queuing theory with experimental validation», *IEEE Transactions on Mobile Computing*, vol. 20, n.º 4, págs. 1558-1573, 2021.

[111] W. Whitt, «The queueing network analyzer», *The bell system technical journal*, vol. 62, n.º 9, págs. 2779-2815, 1983.

[112] S. Agarwal, F. Malandrino, C. F. Chiasserini et al. «VNF placement and resource allocation for the support of vertical services in 5G networks», *IEEE/ACM Transactions on Networking*, vol. 27, n.º 1, págs. 433-446, 2019.

[113] R. Solozabal, J. Ceberio, A. Sanchoyerto et al. «Virtual Network Function Placement Optimization with Deep Reinforcement Learning», *IEEE Journal on Selected Areas in Communications*, vol. 38, n.º 2, págs. 292-303, 2020. DOI: 10.1109/JSAC.2019.2959183.

[114] H. Zorghani, «A Queueing Theory-Based Modeling and Performance Analysis of Push-To-Talk over Cellular Networks», Tesis doct., Florida Institute of Technology, 2019.

[115] W. Garey, T. R. Henderson, Y. Sun et al. «Modeling MCPTT and User Behavior in ns-3.», en *SIMULTECH*, 2021, págs. 30-41.

[116] R. Solozabal, A. Sanchoyerto, E. Atxutegi et al. «Exploitation of mobile edge computing in 5G distributed mission-critical push-to-talk service deployment», *IEEE Access*, vol. 6, págs. 37 665-37 675, 2018.

[117] C. Rotter y T. Van Do, «A queueing model for threshold-based scaling of UPF instances in 5G core», *IEEE Access*, vol. 9, págs. 81 443-81 453, 2021.

[118] C.-Y. Hsieh, T. Phung-Duc, Y. Ren et al. «Design and analysis of dynamic block-setup reservation algorithm for 5G network slicing», *IEEE Transactions on Mobile Computing*, 2022.

[119] S. El Kafhali y K. Salah, «Performance analysis of multi-core VMs hosting cloud SaaS applications», *Computer Standards & Interfaces*, vol. 55, págs. 126-135, 2018.

[120] ——, «Performance modelling and analysis of Internet of Things enabled healthcare monitoring systems», *IET Networks*, vol. 8, n.º 1, págs. 48-58, 2019.

[121] I. Keramidi, D. Uzunidis, I. Moscholios et al. «On Queueing Models for the Performance Analysis of a Vehicular Ad Hoc Network», en *2022 International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, IEEE, 2022, págs. 1-6.

[122] G. Faraci, A. Lombardo y G. Schembra, «A building block to model an SDN/NFV network», en *2017 IEEE International Conference on Communications (ICC)*, IEEE, 2017, págs. 1-7.

[123] L. Zabala, A. Ferro y A. Nieva, «Modelo de colas con vacations aplicado a un sistema de captura de paquetes», *XIII Jornadas de Ingeniería Telemática (JITEL 2017). Libro de actas*, págs. 132-139, 2018.

[124] F. Schneider, J. Wallerich y A. Feldman, «Packet Capture in 10-Gigabit Ethernet Environments Using Contemporary Commodity Hardware», en *Proceedings of the 8th International Passive and Active Measurement Conference, PAM 2007,Louvain-la-neuve, Belgium, April 5-6*, Springer, 2007, págs. 207-217.

[125] D. Bovet y M. Cesati, *Understanding the Linux Kernel, Third Edition*. O'Reilly Media, 2005.

[126] W. Leland, M. Taqqu, W. Willinger et al. «On the self-similar nature of Ethernet traffic», *IEEE/ACM Transactions on Networking*, vol. 2, n.º 1, págs. 1-15, 1994.

[127] J. R. Norris, *Markov chains*, 2. Cambridge university press, 1998.

[128] M. L. Puterman, *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[129] D. Bertsekas, *Dynamic programming and optimal control: Volume I*. Athena scientific, 2012, vol. 1.

[130] A. Pineda, L. Zabala y A. Ferro, «Network architecture to automatically test traffic monitoring systems», en *Proceedings of the Mosharaka International Conference on Communications and Signal Processing (MIC-CSP2012), Barcelona, Spain*, Mosharaka, 2012.

[131] E. Freitas, A. T. de Oliveira Filho, P. R. do Carmo et al. «A survey on accelerating technologies for fast network packet processing in Linux environments», *Computer Communications*, 2022.

[132] C. Benvenuti, *Understanding Linux network internals*. O'Reilly Media, Inc., 2006.

[133] C. A. Petri, «Kommunikation mit automaten», 1962.

[134] T. Murata, «Petri nets: Properties, analysis and applications», *Proceedings of the IEEE*, vol. 77, n.º 4, págs. 541-580, 1989.

[135] W. J. Thong y M. Ameedeen, «A survey of Petri net tools», en *Advanced Computer and Communication Engineering Technology*, Springer, 2015, págs. 537-551.

[136] V. B. Kumbhar y M. S. Chavan, «A Review of Petri Net Tools and Recommendations», en *International Conference on Applications of Machine Intelligence and Data Analytics (ICAMIDA 2022)*, Atlantis Press, 2023, págs. 710-721.

[137] R. J. Rodriguez, S. Bernardi y A. Zimmermann, «An Evaluation Framework for Comparative Analysis of Generalized Stochastic Petri Net Simulation Techniques», *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2018.

[138] A. Zimmermann, «Modelling and Performance Evaluation with TimeNET 4.4», en *International Conference on Quantitative Evaluation of Systems*, Springer, 2017, págs. 300-303. DOI: 10.1007/978-3-319-66335-7_19.

[139] J. H. Salim, R. Olsson y A. Kuznetsov, «Beyond Softnet», en *Annual Linux Showcase & Conference*, vol. 5, 2001, págs. 18-18.

[140] N. Tian y Z. G. Zhang, *Vacation queueing models: theory and applications*. Springer Science & Business Media, 2006, vol. 93.

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

[142] D. Fiems, «Analysis of discrete-time queueing systems with vacations», Tesis doct., Ghent University, 2004.

[143] *The Linux Kernel Documentation*, https://docs.kernel.org, accessed: 2023-02-24.

[144] G. Bolch, S. Greiner, H. De Meer et al. *Queueing networks and Markov chains: modeling and performance evaluation with computer science applications*. John Wiley & Sons, 2006.

[145] I. Adan y J. Resing, *Queueing theory*, 2002.

[146] R. B. Cooper, «Queueing theory», en *Proceedings of the ACM'81 conference*, 1981, págs. 119-122.

[147] T. T. Lee, «M/G/1/N queue with vacation time and exhaustive service discipline», *Operations Research*, vol. 32, n.º 4, págs. 774-784, 1984.

[148] 3GPP TS 23.379, *Functional architecture and information flows to support Mission Critical Push To Talk (MCPTT); Stage 2 (Release 14)*, ver. 14.5.0, abr. de 2018.

[149] J. J. Pazos Arias, A. Suárez González y R. P. Díaz Redondo, *Teoría de colas y simulación de eventos discretos*. Pearson-Prentice Hall, 2003.

[150] 3GPP TS 22.179, *Technical Specification Group Services and System Aspects; Mission Critical Push to Talk (MCPTT) over LTE; Stage 1*, ver. 14.3.0, dic. de 2016.

[151] T. T. Lee, «M/G/1/N queue with vacation time and limited service discipline», *Performance Evaluation*, vol. 9, n.º 3, págs. 181-190, 1989.

[152] A. Laghrissi y T. Taleb, «A survey on the placement of virtual resources and virtual network functions», *IEEE Communications Surveys & Tutorials*, vol. 21, n.º 2, págs. 1409-1434, 2018.

[153] W. Attaoui, E. Sabir, H. Elbiaze et al. «VNF and Container Placement: Recent Advances and Future Trends», *arXiv preprint arXiv:2204.00178*, 2022.

[154] E. Ibarrola, E. Saiz, L. Zabala et al. «A New Global Quality of Service Model: QoXphere», *IEEE Communications Magazine*, vol. 52, n.º 1, págs. 193-199, 2014. DOI: 10.1109/MCOM.2014.6710083.