Universidad del País Vasco Euskal Herriko Unibertsitatea
BILBOKO INGENIARITZA ESKOLA
ESCUELA DE INGENIERÍA DE BILBAO

GRADO EN INGENIERÍA EN TECNOLOGÍA DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

SOLUCIÓN PARA RESPUESTA ANTE FALLOS DE ENLACE EN REDES DE COMUNICACIONES, BASADA EN DPDK Y ANÁLISIS DE RENDIMIENTO

Estudiante: López Alejandro, Sergio

Director/Directora: Higuero Aperribay, María Victoria

Codirector/Codirectora:

Curso: 2023-2024

Fecha: Bilbao, 27 de Junio de 2024

Agradecimientos

Querría transmitir mi más sincero agradecimiento a todas las personas e instituciones que me han proporcionado su ayuda para la elaboración de este TFG.

Al grupo I2T por facilitar los medios, la ayuda y el ambiente de trabajo necesarios.

A la directora del proyecto, María Victoria Higuero y a David Franco por su orientación, conocimiento y paciencia durante todo el proyecto.

Finalmente, pero no por ello menos importante, a mi familia y amigos por su compañía a lo largo de estos años de trabajo y aprendizaje.

Resumen

Este Trabajo de Fin de Grado presenta una propuesta de solución ante situaciones de fallos de enlaces en redes de telecomunicaciones, haciendo uso de la tecnología DPDK junto con un análisis de rendimiento de dicha solución. De forma más concreta, el proyecto aprovecha el concepto de virtualización de hardware y la tecnología DPDK para el diseño y desarrollo de una solución que permite el procesado de paquetes y la atención y reacción a caídas de enlace.

Palabras clave: DPDK, SDN, plano de datos programable, procesado de paquetes, optimización de red.

This Bachelor's Thesis presents an alternative proposal to other technologies as a solution to address link failures in telecommunications networks using DPDK technology along with its corresponding performance analysis. More specifically, the project leverages the concept of hardware virtualization and DPDK technology for the design and development of a solution that allows packet processing and response to link failures.

Keywords: DPDK, SDN, programmable data plane, packet processing, network optimization.

Gradu Amaierako Lan hau beste teknologia batzuei alternatiba bat aurkezten die telekomunikazio sareetan esteka-hutsei erantzuteko, DPDK teknologia eta bere errendimendu-analisiarekin batera erabiliz. Zehatzago esanda, proiektuak hardware birtualizazioaren kontzeptua eta DPDK teknologia aprobetxatzen ditu, paketeak prozesatzeko eta esteka-hutsei erantzuteko eta arreta emateko soluzioa diseinatzeko eta garatzeko.

Gako-hitzak: DPDK, SDN, programagarria datu-planoa, pakete-prozesamendua, sare-optimizazioa.

Índice de contenido

Indicé de figuras

Indicé de tablas

1 Introducción

Con el crecimiento de la utilización de equipos conectados a las redes de comunicaciones, los proveedores de servicios de Internet cada vez requieren de infraestructuras que puedan abastecer a todos sus clientes al mismo tiempo que mantienen las redes seguras y con el adecuado nivel de calidad de servicio. Un concepto nacido hace no muchos años con el que se obtendrían ciertas ventajas fue el de las redes SDN (Software Defined Networking). Este nuevo tipo de redes facilita considerablemente la gestión y mantenimiento mientras que gestionan incluso mejor el tráfico que viaja a lo largo de las mismas.

Como un paso más innovador en la evolución de las SDN, y con la intención de realizar una mejora adicional en dichas redes, surge el concepto denominado plano de datos programable. Este nuevo concepto permite una personalización mayor del comportamiento de cada dispositivo de red al mismo tiempo que proporciona velocidades considerables en el procesado y gestión del tráfico. Para la definición de este plano entre varias tecnologías podemos encontrar P4 o DPDK entre otras, las cuales nos permiten no solo esa personalización y una mejora de rendimiento, sino también la posibilidad de realizar un control de tráfico más coordinado.

Este proyecto ha sido desarrollado en el grupo de investigación I2T [1] perteneciente a la Escuela de Ingeniería de Bilbao siguiendo una de sus líneas de investigación centrada en el campo de las redes SDN con PDP (Plano de Datos Programable). El proyecto se ha desarrollado con el fin de definir una solución de procesado de paquetes haciendo uso de las tecnologías mencionadas anteriormente. Además, también se realiza un estudio de su rendimiento frente a otras alternativas más utilizadas hasta la fecha, como podría ser P4.

2 Contexto tecnológico

En este apartado se describen las tecnologías más relevantes relacionadas con este TFG. Dentro de estas, se encuentran tecnologías como las redes SDN (Software Defined Networking), PDP (Programmable Data Plane) y otros, como DPDK y P4. A continuación, se proporciona una descripción de estas tecnologías, incluyendo información sobre sus componentes, su funcionamiento básico y su importancia en el contexto del proyecto.

2.1 Redes SDN

Las redes SDN (Software Defined Networks), nacen como una tecnología alternativa a las redes tradicionales. Se crearon mediante la búsqueda de un tipo de redes que pudiese satisfacer las necesidades de los servicios de red al mismo tiempo que facilitaba otras funcionalidades como mantenimiento y centralización de la gestión de las redes entre otras.

La arquitectura de las SDNs consta de 3 planos de funcionamiento: aplicación, control y datos. La capa de datos principalmente se centra en realizar el encaminamiento del tráfico a través de la red, mediante la configuración definida desde el plano de control. El plano de control es más complejo y la base del concepto de las redes SDN. Esta capa es responsable de aplicar las políticas de dicha red sobre los nodos del plano de datos. Por último, el plano de aplicación es el que contiene los programas y aplicaciones que comunican el funcionamiento requerido de la red a los niveles inferiores mediante una interfaz que comparten.

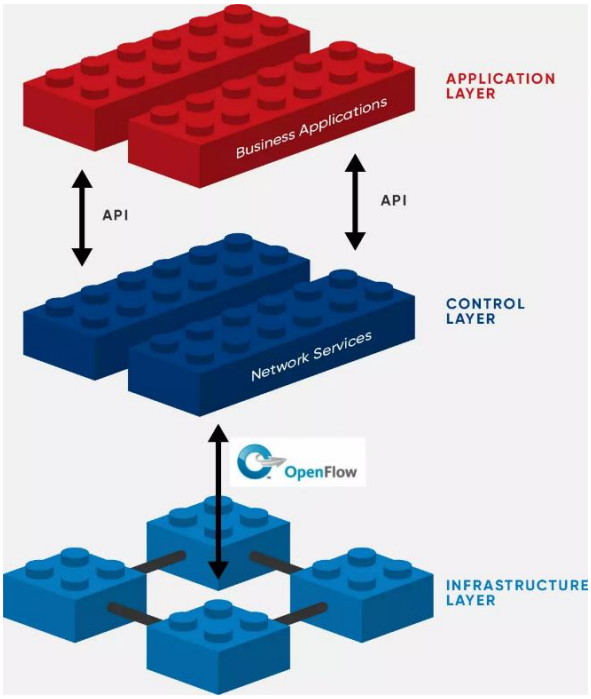

Ilustración 1: SDN Architecture
Source: [2]

La utilización de estas tres capas permite que el plano de datos no tenga que ser responsable de la configuración de encaminamiento mientras que la aplicación junto con los nodos de control permite realizar modificaciones y mantenimiento atendiendo a las políticas de rutado en la red, de una forma más eficiente que en otro tipo de redes. Esa característica proporciona a este tipo de redes una flexibilidad mayor y por tanto facilidad para escalar las redes con el aumento del tráfico.

Las redes SDN dotan de un nivel mayor de automatización a su configuración, ya que posibilitan el hacer cambios de configuración en los nodos en tiempo real según las circunstancias de la red. Por ejemplo, la habilitación de enlaces adicionales dependiendo de la demanda actual o un fowarding más eficaz del tráfico en casos de congestión.

Respecto a la seguridad, esta estructura permite llevar a cabo planes de actuación ante incidencias de forma eficaz manipulando desde el plano de aplicación sin tener que intervenir en los nodos del plano de datos directamente. Este tipo de redes soportan grandes velocidades para el tráfico actual, sin embargo, la escalabilidad, reconfiguración y mantenimiento de los equipos específicos de la red puede ser un problema.

La separación de los niveles de datos y control en las redes SDN facilita la globalización del mantenimiento y configuración de las redes. Este concepto es lo que permite aplicar configuraciones simultaneas a lo largo de toda una red y de forma coordinada. A pesar de ello, se observa excesiva dependencia de dicha capa sobre el plano de datos.

Esto principalmente se debe a que toda toma de decisión respecto al procesado de los paquetes se realiza desde el controlador, los dispositivos en el plano de datos actúan bajo sus órdenes sin ningún tipo de inteligencia. Estas órdenes y decisiones se realizan mediante una comunicación basada en la utilización de un protocolo, como podría ser OpenFlow entre otros. [3]

2.2 Planos de datos programables

Como siguiente paso en las redes SDN, surge el concepto del Plano de Datos Programables (PDP). Esta tecnología consiste en dotar de cierta inteligencia a los equipos de este nivel de datos. Esto permite poder mantener un equilibrio entre el poder facilitar la configuración de los equipos mediante el nivel de control y al mismo tiempo evitar esa dependencia entre los dos niveles.

Además de dicha independencia, hay que destacar una característica de este tipo de redes que es la posibilidad de la implementación de configuraciones de una forma centralizada, lo que permite la unificación de toda la red al mismo tiempo que proporciona dicha programabilidad.

Este modo de funcionamiento permite que se puedan aplicar en cada dispositivo de la red diferentes configuraciones en relación con el procesado de cada mensaje o unidad de datos que llega a cada dispositivo de red. Entre la variedad de implementaciones que se pueden llegar a desplegar podemos encontrar firewalls, balanceadores de carga, aceleración de hardware y QoS entre otros.

Un ejemplo notable de este tipo de implementaciones podría ser HULA[4], el cual basado en P4, que es un lenguaje para la propagación del PDP, es un algoritmo bastante flexible y escalable de balanceo de carga en el plano de datos el cual supera otros sistemas como CONGA siendo este un diseño para redes demasiado específicas como centros de datos.

2.2.1 Aceleración de funciones de red

El concepto de la aceleración de funciones de red podría definirse como un conjunto de modificaciones por parte del hardware y software de los nodos de la red realizadas sobre las operaciones de procesado de paquetes y tráfico con el fin de facilitar y mejorar la gestión de los paquetes en la red.

Un ejemplo de la aplicación de esta tecnología podría tratarse de las ya mencionadas redes SDN, ya que la separación de las distintas responsabilidades de una red en varios niveles permite a cada uno de ellos centrarse en sus funciones. Además de esta característica, encontramos también la opción de aceleración hardware del plano de datos. Es decir, una vez reducidas sus funciones, también se podrá mejorar la recepción y envío de paquetes.

Esto es posible mediante la posibilidad de hacer uso de planos de datos programables (PDP). Es decir, proporcionar un mayor nivel de customización del funcionamiento de los nodos de la red con relación al procesado de paquetes posibilitando así el poder bien evitar procesos innecesarios para ofrecer mayores velocidades en el procesado y envío de tráfico en cada dispositivo o incluso dotarles de mayor inteligencia computacional. Para esta programabilidad en los nodos del Data Plane se podría hacer uso del lenguaje “P4” entre otras tecnologías, el cual permitirá definir en diferentes escenarios de red las funcionalidades requeridas.

2.2.2 P4

Programming Protocol-independent Packet Processors o más conocido como P4 surge en 2013 en la *"P4 Language Consortium"* [5], una organización formada por los grupos Google, Intel, Microsoft Research, Barefoot Networks, Princeton y Stanford. En ese momento se requería de la creación de un estándar que pudiese ser usado para definir la forma en la que se procesaban y redireccionaban los paquetes que pasaban por los nodos de una forma más independiente que en una red SDN más básica. [6] [3]

Este lenguaje de programación permite proporcionar un nivel de personalización mayor en ese tratamiento del tráfico que en las redes tradicionales. Dentro de las funcionalidades a incorporar destacamos un forwarding más adaptado a cada situación, balanceo de carga, implementación de políticas QoS, entre otras.

Según sus creadores [7], podemos destacar varias características:

1. Flexibilidad: P4 posibilita el poder implementar diferentes políticas de forwarding mediante su programación.
2. Expresividad: La programación permite expresar algoritmos complejos usando únicamente operaciones simples y búsquedas de tablas.
3. Disponibilidad de librerías: Pueden hacerse uso de librerías definidas por los fabricantes para realizar funciones específicas para dicho hardware.
4. Soporte: Los desarrolladores pueden proporcionar software específico para ciertas redes con el fin de ayudar en el desarrollo de P4.

Como consecuencia de la estandarización de este lenguaje y en relación con la mejora de la facilidad de su uso, surgieron creaciones como *"Barefoot Tofino"* [8], un chip programable en P4 que alcanza grandes velocidades de transmisión, "P4Runtime" [9] el cual es una especificación del plano de control o *"Stratum Project"* [10], un sistema operativo diseñado para switches en redes SDN.

2.2.3 DPDK

DPDK (Data Plane Development Kit), es un conjunto de librerías desarrolladas por Intel. Este Kit tiene como objetivo el proveer de un framework que proporciona la posibilidad de realizar algunas operaciones de procesado de paquetes de aplicaciones en el plano de datos. [11]

2.2.3.1 Objetivo de DPDK

Para describir el papel de DPDK dentro del plano de datos, se describe brevemente el procesado habitual de un paquete por parte de un equipo cuando se recibe a través de una interfaz de red.

En un proceso habitual de recepción de un paquete que llega a una NIC (Network Interface Card), el paquete es obtenido por el Kernel y ahí se descomprimen sus cabeceras de nivel de enlace, red y transporte. Una vez obtenido el contenido del nivel de aplicación, se guarda en un espacio de memoria en el cual solo el Kernel tiene acceso, generalmente denominado como *Kernel space* y posteriormente se hace una copia de este contenido al espacio de memoria accesible por el programa al que le corresponde dicho paquete.

DPDK, permite obtener directamente el paquete completo del NIC y tenerlo a disposición del programa desarrollado mediante el uso de dichas librerías. Al hacerlo, se dispone de la capacidad para realizar cualquier operación necesaria sobre dicho paquete. Dentro de estas modificaciones, podríamos destacar la comprobación del *Checksum*, modificación de cabeceras o encapsulamiento, entre otras. Es importante destacar que al obtener el paquete completo se deben realizar las operaciones que se requieran, que no están siendo realizadas por el Kernel ya que por defecto no hay ningún tipo de control de si los paquetes recibidos están en correctas condiciones para ser procesados

Se debe tener en cuenta que, considerando la reducción del tiempo de procesado en cada nodo debido a la omisión de los pasos mencionados anteriormente llevados a cabo por el Kernel, se produce una mejora considerable en el tiempo de procesado de paquetes individualmente y de la velocidad del tráfico en general. Además, permite la personalización de dichos paquetes de una forma considerablemente eficiente.

2.2.3.2 Funcionamiento DPDK

Los sistemas operativos disponen de cierta estructura jerárquica en la cual se sitúan los diferentes componentes software dependiendo del nivel de sus privilegios con relación a los diferentes componentes de un equipo. Dichos niveles de privilegios se representan mediante el conocido *"Protection ring"*, Siendo el núcleo el Kernel, y el anillo exterior las aplicaciones de usuario.

Para el caso de DPDK, su ejecución se sitúa en el nivel exterior de dicho anillo; es decir, no debería de disponer de los permisos para acceder al hardware de un equipo. Sin embargo, mediante el concepto de virtualización de hardware sí es posible permitir a un programa DPDK acceder a cierto hardware especifico.

Cabe destacar que DPDK no tendría los mismos permisos de acceso que una aplicación ejecutándose a nivel de Kernel, pero dicha virtualización sería suficiente para la realizar funcionalidades como la recepción, procesado y envío de paquetes, o la configuración y negociación de parámetros indispensables para una comunicación.

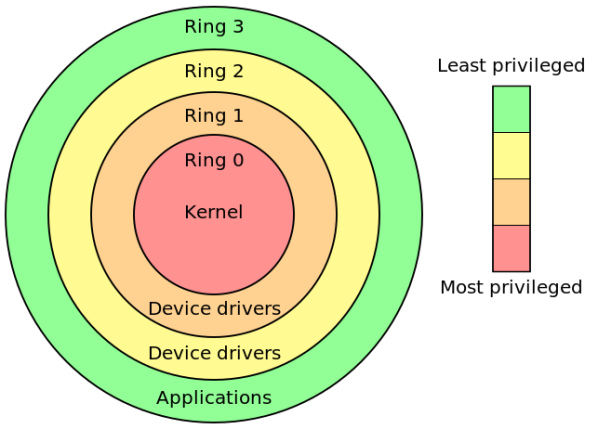

Ilustración 2: Protection Ring
Source: [12]

Por defecto, el software con los permisos correspondientes al anillo exterior no debería tener acceso a ningún tipo de interrupciones de hardware. Sin embargo, esto es posible mediante el concepto "Virtual Function" (VF) [13]. La tecnología SR-IOV (Single Root I/O Virtualization), permite que un dispositivo I/O como una tarjeta de red (NIC) en este caso pueda ser virtualizada en instancias que se denominan funciones virtuales. Estas funciones pueden ser accesibles por una máquina virtual, un contenedor, etc.

Se destacan UIO (Userspace I/O) y VFIO (Virtual Function I/O) como los drivers más usados para dicha funcionalidad. Para su uso con DPDK se suele recomendar usar VFIO ya que, a diferencia de su alternativa, este permite la reserva de la interfaz proporcionándole acceso completo a DPDK y evitando que cualquier otro proceso externo a DPDK intente hacer uso de las interfaces virtualizadas.

El Poll Mode Driver (PMD) [14] es otro aspecto esencial para el funcionamiento de DPDK. Este controlador es ejecutado en el nivel de usuario, se encarga principalmente en realizar llamadas periódicas al NIC para obtener los paquetes recibidos. Es decir, DPDK no espera a que un paquete llegue al NIC, detectar la interrupción y solicitar el paquete, simplemente con esa periodicidad realiza comprobaciones constantes. Es importante destacar de este concepto que con el fin de mantener un funcionamiento óptimo la frecuencia a la que realiza dichos sondeos se aumenta o reduce según las necesidades del tráfico actual.

Esta característica principalmente ayuda a mejorar las velocidades de obtención y entrega de paquetes, ya que este evita el uso de atención a interrupciones por parte de la CPU en el NIC (con excepción de las interrupciones producidas por el cambio del estado del enlace, como desconexiones). En la siguiente figura (Ilustración 3), se muestra la comparación entre un procesado estándar el cual se realiza en el Kernel con el que ocurre al hacer uso de la libraría DPDK, donde dicho procesamiento se produce a nivel de la aplicación.

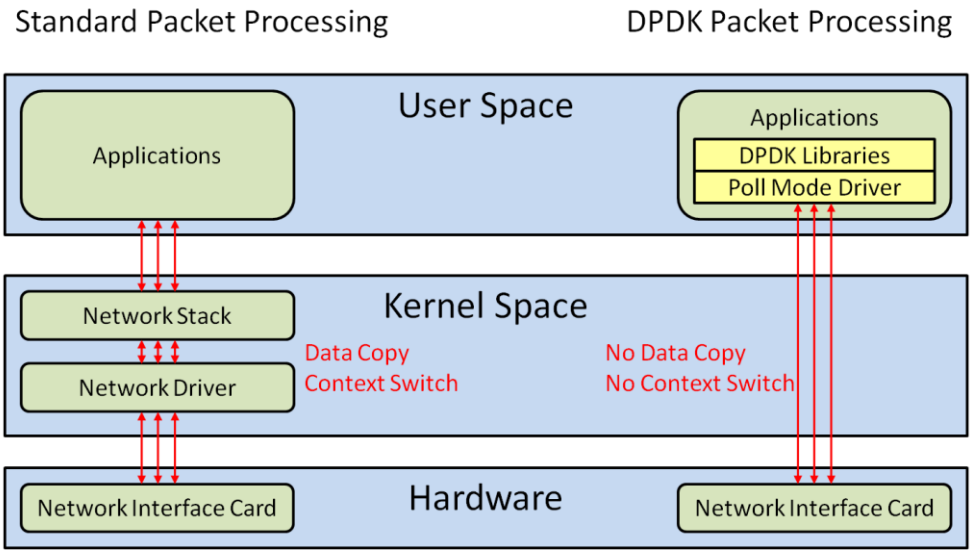

Ilustración 3: Standard vs DPDK Packet Processing
Source: [15]

3 Objetivos y alcance

El objetivo principal de este proyecto consiste en el diseño e implementación de una solución de red basada en DPDK (Data Plane Development Kit) para dar una respuesta eficiente ante una caída de enlace. Además, se realizarán diversas pruebas de rendimiento con el fin de comprobar el correcto funcionamiento de la solución. Dicho objetivo se llevará a cabo mediante las siguientes metas.

- Diseño de la solución de red: Desarrollo de una solución robusta que soporte la recepción y transmisión de datos de forma eficiente al mismo tiempo que minimiza el tiempo de reacción ante un fallo de enlace con el fin de que dicha situación tenga el menor impacto posible en el servicio que perciben los usuarios finales.

- Implementación de respuesta ante caídas de enlace: Implementación del algoritmo correspondiente para la detección de la caída de un enlace al mismo tiempo que se toman las medidas necesarias para recuperar la conectividad mediante el establecimiento de una nueva ruta alternativa en el menor tiempo posible garantizando retardos y perdidas de datos mínimos durante el evento.

- Análisis y evaluación de rendimiento: Evaluación del funcionamiento de la solución diseñada e implementada mediante un plan de pruebas adecuado para el análisis de rendimiento. Todo ello documentado y presentado de forma detallada realizando las comparaciones necesarias para su evaluación.

4 Beneficios del proyecto

El desarrollo de este proyecto podrá beneficiar considerablemente a diferentes aspectos en la sociedad, tecnología y economía. A continuación, se muestran dichos beneficios justificando el motivo de su mención y la forma en la que se podrían alcanzar.

4.1 Beneficios sociales

Conexión para servicios críticos:

Uno de los principales beneficios a la sociedad derivados del desarrollo de esta solución es el dotar a una red de telecomunicaciones en centros de servicios críticos o de urgencia de características que minimice el impacto de los problemas producidos por una desconexión , en las comunicaciones de extremo a extremo.

En concreto, la característica de la utilización de enlaces alternativos con un tiempo de recuperación y perdida de paquetes mínimos puede evitar que ciertos servicios pierdan datos. Este concepto es esencial en situaciones en las que la disponibilidad de la red puede marcar una diferencia en la vida de muchas personas como en hospitales, servicios de emergencia y centros de seguridad. La característica de proporcionar un servicio de comunicaciones constante y confiable permite que dichos servicios funcionen de forma eficiente.

Conectividad constante áreas remotas:

Para el caso de usuarios situados en áreas remotas, una ventaja considerable es el tener a disposición una solución que aproveche el ancho de banda disponible. Además, el concepto de tener a disposición enlaces alternativos evita realizar un mantenimiento de equipamiento físico en áreas de difícil acceso. En caso de necesitar realizar dicho mantenimiento el enlace alternativo proporcionara tiempo suficiente para coordinar y realizar dicha revisión sin producir ningún tipo de interrupción en el servicio.

Considerando las características mencionadas, esta solución podría destacar también en regiones rurales y aisladas en las que hay una infraestructura de telecomunicaciones limitada y debido a las condicione geográficas el acceso es más complicado.

4.2 Beneficios tecnológicos

Optimización del rendimiento de la red:

Considerando la importancia de las redes SDN hoy en día, es fácil suponer que la implementación de este tipo de soluciones en redes de telecomunicaciones de forma general proporcionaría una mejora significativa.

En concreto, el escenario en el que se dispone de un gran número de nodos interconectados entre si haciendo uso de DPDK junto con el concepto de recuperación ante caídas de enlaces, puede crear una red más robusta y con una recuperación más rápida de la red.

Escalabilidad y flexibilidad:

La solución diseñada permite la incorporación de nuevos nodos a la red con su configuración propia y con el número de interfaces que se requiera acorde con las características del equipo y la necesidad en la red.

4.3 Beneficios económicos

Coste por interrupción:

Teniendo en cuenta la cantidad de información que se intercambia entre las grandes empresas por sus enlaces de sus centros de datos dando servicio a millones de personas al día, el sufrir una interrupción en sus comunicaciones de unos pocos segundos puede generar una gran cantidad de pérdidas económicas.

Es por ello por lo que la implementación de este tipo de soluciones podría suponer una alternativa para tener en cuenta. Principalmente por poder proveer de grandes velocidades con el hardware apropiado al mismo tiempo que se reduce considerablemente la perdida de paquetes por caída de enlaces de forma eficiente.

Aumento competitividad y alternativa:

Por último, independientemente de los beneficios que este tipo de soluciones puedan proporcionar, la competitividad que genera esta solución ante otras alternativas podrá ayudar de una manera global la mejora de aquellas tecnologías más desactualizadas o la sustitución por aquellas que se adecuan correctamente a las necesidades de cada red.

5 Análisis de alternativas

A lo largo del análisis de alternativas se presentan todos los componentes que se han valorado a la hora de diseñar, implementar y testear la solución presentada. Para cada toma de decisión se han tenido en cuenta diferentes criterios con sus correspondientes ponderaciones facilitando así la toma de decisiones. Además, también se exponen los diferentes motivos por los que se ha valorado cada uno de los componentes y el porqué de su puntuación.

5.1 Equipamiento

A lo largo de la realización del plan de pruebas se ha valorado que tipo de infraestructura sería la más conveniente para realizar la demostración de la solución desarrollada. Para su evaluación se han tenido en cuenta los siguientes criterios:

- Configuración: Indica la complejidad con la se debe lidiar para configurar la arquitectura centrándose en aspectos como la instalación de software, comunicación entre equipos, evitando así un exceso de tiempo y coste del proyecto.

- Coste: Dado que el equipamiento físico suele tener un coste considerable respecto al virtual se ha incluido este criterio valorando así con 1 coste muy alto y 5 coste bajo.

- precisión: Este criterio se centra en el análisis de como de cercano estaría la arquitectura seleccionada con una situación real. Considerando la relevancia de la interconexión de equipos, el cableado entre otros factores podría ser relevantes a la hora de diseñar la solución.

- Compatibilidad: Evalúa como de compatible es la tecnología usada con el equipamiento seleccionado en cada uno de los escenarios analizados. Esto se debe a ciertos conflictos que puedan producirse con la virtualización de la interfaces de red.

Tabla 1:Tabla de decisión Equipamiento

	Configuración (30%)	Coste (20%)	Precisión (20%)	Compatibilidad (30%)	TOTAL
Maqueta virtual	4	4	2	3	3.3
Máquina virtual	4	3	4	3	3.5
Equipamiento real	3	2	5	5	3.8

El uso de una maquita completamente virtual es una opción para valorar a la hora de realizar la implementación de cualquier solución de red. Sin embargo, para el equipamiento actual ha sido de importancia disponer de al menos dos tarjetas de red físicas. Esto se debe a que, para la configuración, maniobrabilidad de la interconexión entre los equipos y la realización de las pruebas correspondientes, tener a disposición el cableado físico facilita estas funciones considerablemente. El coste y la configuración son aspectos bastante positivos por los que valorar

esta arquitectura; sin embargo, no son lo suficiente relevantes para considerarlo como la mejor opción.

Como una alternativa se han considerado las máquinas virtuales. Estas tendrían un coste razonable al tener que usar al menos dos equipos, pero es cierto que los equipos finales podrían incluirse en dichos equipos al mismo tiempo que se tiene acceso a la interconexión entre equipos. Sin embargo, dada su compatibilidad más compleja respecto a la virtualización de la tarjeta de red y su menor precisión respecto a un equipamiento físico, no es suficiente.

Por último, hacer uso de equipamiento real proporciona un análisis mucho más cercano a un escenario real. Puede que su coste se incremente bastante respecto a las arquitecturas anteriores, sin embargo, dado que esta tecnología se centra en la virtualización de hardware y ciertas manipulaciones en el Kernel, el disponer de equipos por separado para cada funcionalidad incrementa su compatibilidad con la virtualización en sí.

Teniendo en cuenta la disposición de un portátil personal, el equipamiento disponible del laboratorio del grupo I2T, y las características más adecuadas de la arquitectura, se ha decidido hacer uso de equipamiento real para esta solución.

5.2 Sistema operativo

Considerando la variedad de sistemas operativos disponibles y tratándose de una solución de red, es esencial el valorar que sistema operativo utilizar en los diferentes equipos de la infraestructura. A continuación, se presentan los criterios utilizados y brevemente los motivos de su selección.

- Soporte: Se evalúa como de frecuente o eficaz es el soporte y mantenimiento del sistema operativo. Esta característica se ha seleccionado debido a la importancia de evitar el cambio del sistema operativo de los equipamientos en el futuro por falta de actualizaciones, sobre todo de seguridad.

- Nivel de utilización: No es una característica esencial, pero si puede facilitar el encontrar información sobre el sistema operativo con respecto a posibles errores o incluso facilita su mantenimiento al ser más conocido.

- Estabilidad: El poder mantener el sistema operativo en funcionamiento durante largos periodos de tiempo, y siendo manipulado constantemente sin que sufra ninguna variación excesiva en el rendimiento. Es un parámetro relevante al tratarse de una solución para el reenvío de paquetes.

- Compatibilidad: Este criterio se centra en analizar de forma general como de compatible es cada sistema operativo con la tecnología seleccionada e incluso con el hardware, ya que en caso de requerir un escalado o migración de la red se evitan un gran número de conflictos.

Tabla 2: Tabla de decisión Sistema Operativo

	Soporte (20%)	Popularidad (15%)	Estabilidad (25%)	Compatibilidad (25%)	TOTAL
Windows	3	5	3	4	3,7
Linux - Ubuntu	4	4	4	5	4,25
Linux – Otras distribuciones	4	4	3	4	3,75
FreeBSD	4	3	4	4	4

Mediante la ponderación de las características seleccionadas se puede observar que Ubuntu sería el más apropiado siendo uno de los sistemas con una estabilidad decente, un buen soporte y principalmente su compatibilidad con la mayoría del hardware disponible en el mercado. FreeBSD se presentaba como una prepuesta bastante interesante sin embargo considerando que una de sus características más conocidas es la división de su Kernel en módulos, no parece ser una característica adecuada para esta solución.

Es por ello por lo que para el caso de los dos equipos principales usados en la maqueta se realiza la instalación de Ubuntu. Por otro lado, en el equipo portátil de pruebas se combina el uso de Windows junto con el sistema Kali Linux para realizar las pruebas de rendimiento por su facilidad de uso y variedad de herramientas disponibles.

5.3 Software de pruebas

Respecto al software a utilizar durante las pruebas se han analizado las opciones incluidas a continuación junto con sus criterios correspondientes:

- Coste: Criterio seleccionado con el fin de evitar la compra de software teniendo en cuenta la disponibilidad de alternativas gratuitas.

- Estabilidad: Al tratarse de una solución en la cual pequeñas diferencias de tiempo de nanosegundos tienen relevancia en su estudio, es esencial el disponer de un software que evite grandes variaciones de rendimiento.

Tabla 3: Tabla de decisión Software de pruebas

	Coste (30%)	Estabilidad (40%)	TOTAL
Ping	5	5	5
iperf	5	4	4,3
NetScanTools	1	5	3,2

Observando la Tabla 3, se puede apreciar que el uso del protocolo ICMP (Internet Control Message Protocol) mediante la utilidad *Ping* sería la más recomendada. Sin embargo, no se trata de un programa excesivamente complejo y no dispone de una variedad de opciones. Es por ello por lo que se toma a la decisión de complementar la fase de pruebas y rendimiento tanto con Ping como con iperf. Esta última consiste en una conocida herramienta para la medida de una variedad de parámetros en redes de comunicaciones.

6 Análisis de riesgos

Se hace uso del análisis de riesgos con el fin de realizar un estudio detallado de los posibles riesgos que se puedan producir a lo largo del diseño y desarrollo de la solución planteada. Este análisis incluye la identificación de riesgos y evaluación de su probabilidad de ocurrencia. Además, se incluye dos planes clave: un plan preventivo y un plan de contingencia.

El plan preventivo trata de buscar la forma más adecuada para evitar que se produzca un riesgo. Por otro lado, el plan de contingencia se diseña con el fin de evitar el mayor número de consecuencias posible.

6.1 Identificación de riesgos

A continuación, se realiza la identificación de los riesgos tanto externos como internos que se deben considerar en este proyecto. Además, se incluye una breve explicación mencionando en qué consisten dichos riesgos, justificando su análisis.

Riesgos externos

- (1) Ciberataque

- Consiste en el robo, alteración o manipulación de equipamiento o información sin ningún tipo de autorización, de manera forzosa. Este evento es aparentemente poco probable pero ciertamente se producen un gran número de intentos de ataques a redes, sobre todo en instituciones gubernamentales y universitarias entre otras.

  La consecuencia principal de un ataque de este tipo sería la pérdida de toda la información recopilada y generada durante el diseño y desarrollo. En el peor escenario posible, podrían incluso verse comprometidas las credenciales de los repositorios y cuentas de documentación utilizadas.

- (2) Suministro de componentes

  Por haber seleccionado un equipamiento e infraestructura físico para el análisis de alternativas, es posible que sucedan problemas con el suministro de componentes bien por los disponibles en el laboratorio de trabajo, o incluso por la imposibilidad de pedir equipamiento nuevo por presupuesto o tiempo.

  El impacto de este acontecimiento seria considerable ya que no todo el hardware disponible es compatible con DPDK por la necesidad de realizar su virtualización. Además, en caso de que finalmente se obtenga el equipamiento necesario, su impacto podría variar dependiendo de en qué etapa del proyecto se tengan disponibles para su uso.

Riesgos internos

- (3) Errores producidos por actualizaciones de software:

  Uno de los conflictos más comunes en el desarrollo de software es el realizar actualizaciones del software con el que se está trabajando. Especialmente actualizaciones relativas al sistema operativo ya que en muchas ocasiones se actualizan herramientas que se están utilizando y que cambian su funcionamiento y compatibilidad.

  Las consecuencias de este riesgo podrían significar la instalación de todo el sistema operativo y herramientas nuevamente con sus correspondientes versiones, cerciorándose de que toda la configuración inicial realizada se replica correctamente.

- (4) Errores en el código implementado

  Es posible que bien por errores humanos o por desinformación se produzcan ciertos errores en el desarrollo de la solución. La probabilidad de que suceda debido a un error de compilación es excesivamente baja, pero ello no evita que puedan producirse de otro tipo.

  En concreto nos podríamos referir a código implementado que por su excesiva complejidad o el incorrecto uso de diversas funciones afecte al rendimiento de la aplicación. En algunos tipos de soluciones software podría parecer que no es relevante, pero para el caso concreto de manipulación y forwarding de paquetes a altas velocidades podría ser un inconveniente.

6.2 Evaluación probabilidad e impacto

En el siguiente apartado se expone la identificación de la probabilidad y el impacto de cada uno de los riegos mencionados anteriormente. Para su representación se utilizará la tabla conocida como Tabla de Probabilidad-Impacto en la que se puntuará cada uno de los riesgos según su Probabilidad de ocurrencia y su impacto en el proyecto.

Para la consideración de como de relevante es tener en cuenta cada uno de los riesgos, se clasificarán en tres grupos según su puntuaje, de 0.02 a 0.1 de relevancia baja, de 0.08 a 0.25 de relevancia media y finalmente de 0.4 a 0.72 de relevancia alta.

Tabla 4: Tabla Probabilidad Impacto

Probabilidad \ Impacto	Bajo (0,1)	Medio (0,5)	Alto (0,9)
Baja (0,2)	0,02	0,1	0,18 (1)
Media (0,5)	0,05	0,25 (2)	0,45 (3)
Alta (0,8)	008 (4)	0,4	0,72

Los riesgos Ciberataque (1) y Errores por actualización de software (3) se han situado con un impacto alto ya que en ambos casos se requiere la reinstalación de los sistemas operativos y herramientas utilizadas.

Respecto a su probabilidad se ha evaluado que considerando que la red del laboratorio en el que se trabaja tiene una seguridad de nivel alto la probabilidad de que ocurra un ataque satisfactorio (1) es bastante baja. En cambio, una actualización errónea (3). Ya que principalmente podría ocurrir por un error humano, tiene una probabilidad ligeramente más alta (alrededor de 0.25).

La no disposición de hardware (2) se ha situado en una probabilidad media (aproximadamente 0.3) ya que este equipamiento suele tener un coste, y en este caso al requerir unas características concretas podría ocurrir.

Sin embargo, respecto a cuál sería el impacto de no disponer de estos elementos se ha valorado un 0.25 ya que se analizará en el plan de contingencia, se podría estudiar el uso de una de las alternativas mencionadas respecto a la infraestructura utilizada.

Por último, se ha considerado que el impacto producido por errores de código, concretamente los no producidos por compilación, no suponen un riesgo excesivo para el proyecto, a pesar de que puedan generar pequeños problemas de funcionamiento y rendimiento. Sn embargo la probabilidad de ocurrencia es superior a otros riesgos.

6.3 Plan de prevención

En este apartado se presentan los planes de prevención para que sucedan los riesgos con mayor probabilidad o mayor impacto analizados para este proyecto y minimizar su impacto. En este plan únicamente se tendrán en cuenta los riesgos internos ya que los externos se deben a variables que no se pueden controlar desde el proyecto.

- (3) Errores producidos por actualizaciones de software

  Para la prevención de este tipo de errores es importante desactivar cualquier tipo de actualización automática que se pueda prever que pueda generar grandes cambios en las herramientas utilizadas.

  La gran mayoría de sistemas operativos para mayor seguridad permiten el deshabilitar las actualizaciones genéricas y únicamente permitir las relacionadas con la seguridad, ya que estas últimas son muy improbables que afecten al proyecto.

- (4) Errores en el código implementado

  A pesar de que no es un riesgo con un impacto excesivamente alto, es esencial prevenirla mediante revisiones de código durante y en la finalización del desarrollo. Esta prevención en concreto evitará que las consecuencias de este tipo de errores se encuentren durante el periodo de implementación del plan de pruebas.

6.4 Plan de contingencia

Durante el plan de contingencia se mencionarán los procedimientos a seguir en caso de que ocurran tanto los riesgos externos como los internos con el fin de minimizar lo máximo posible su impacto en el proyecto.

- (1) Ciberataque

  La acción principal para contener un posible ataque seria desde la administración de los nodos en la red que estamos trabajando. En caso de no disponer di dichos permisos, se procede a la limpieza completa de los equipos que se han utilizado en la red y siguiendo el correspondiente manual de instalación se recuperara el estado inicial de las maquinas.

- (2) Suministro de componentes

  Aten una falta de equipamiento físico disponible, se valorará un cambio de maqueta para la realización de las pruebas. Como alternativa secundaria se tiene en cuenta el uso de máquinas virtuales pudiendo así usar un único equipo. Sin embargo, se deberá tener en cuenta un posible retraso temporal por la complejidad de la instalación y puesta en marcha en un entorno virtual.

- (3) Errores producidos por actualizaciones de software:

  Para el caso de un error de actualización accidentada, el procedimiento más seguro podría ser el equivalente al de un ciberataque. Sin embargo, con el fin de ahorrar el tiempo disponible para el proyecto, se puede proceder a reinstalar las herramientas que se sospeche que puedan estar interfiriendo por sus versiones originales.

- (4) Errores en el código implementado

  En caso de percibir algún problema de rendimiento que pueda se producido por un problema de código bien durante el desarrollo o incluso durante el plan de pruebas, se podrá proceder con cierta penalización de tiempo en reajustar el código con las modificaciones necesarias.

7 Descripción de la solución

7.1 Introducción

La solución planteada en este proyecto se basa en el hacer uso del concepto de Plano de Datos Programables para el diseño e implementación de un sistema de atención ante caídas de enlace. Como ya se ha mencionado en 2. Contexto tecnológico, existen diversas tecnologías aplicables en este tipo de redes, pero para este caso específico se utiliza la librería DPDK.

En concreto, esta solución se implementa en un nodo que se ha denominado *"Node"*. El cual deberá realizar el reenvío de los paquetes entrantes a través de cualquiera de sus puertos a su salida correspondiente según la dirección MAC destino de dichos paquetes.

Además, dicho nodo debe gestionar cualquier caída de enlace mediante la utilización de un puerto alternativo durante el tiempo en el que se produzca dicho incidente. Una vez que el enlace caído se haya recuperado, el nodo retoma el funcionamiento inicial.

7.2 Arquitectura y escenario

Como se puede apreciar en la Ilustración 3, el equipo "S*ender"* será el que enviar el tráfico hacia el nodo DPDK. Este dispone de dos cables de red conectados a sus dos interfaces junto con las dos interfaces del servidor. Por último, el servidor es el encargado de enviar el tráfico al equipo final correspondiente, este equipo final no será parte de la arquitectura en la realización de las pruebas.

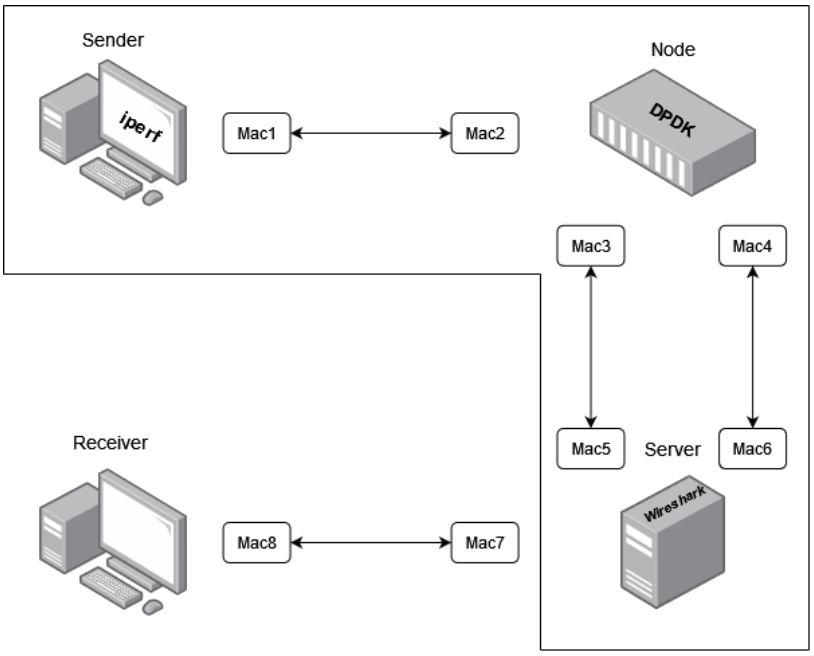

Ilustración 4: Arquitectura

Durante el funcionamiento básico, el nodo se encargaría de direccionar el tráfico según su dirección destino a nivel de enlace. Para el caso de esta pequeña arquitectura, se usa únicamente el primer enlace como se indica en la Ilustración 4.

El nodo al recibir un paquete por su puerto Mac2 comprobará en su tabla de direcciones la MAC destino del paquete y lo enviará por el puerto correspondiente, por defecto en este caso será saliendo por la interfaz Mac3. El enlace alternativo quedaría inactivo.

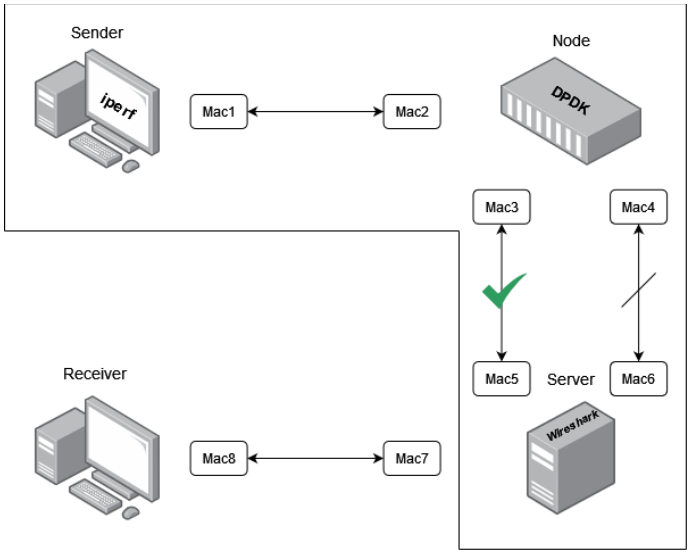

Ilustración 5: Funcionamiento Normal

Cuando se produzca la caída de enlace, DPDK recibirá la interrupción producida por dicha caída y procederá a realizar los cambios necesarios para facilitar el paso del tráfico a través de este. Para ello la interfaz Mac3 se marcará como inactiva, y en la tabla de forwarding se modifican las entradas correspondientes a las direcciones con el puerto Mac3 como salida intercambiándolas por la interfaz Mac4.

El tráfico fluirá con normalidad hasta que se produzca la interrupción indicando que el enlace principal vuelve a estar activo. El procedimiento para este caso sería modificar la tabla de direcciones a su estado anterior y así poder enviar los paquetes por su enlace primario.

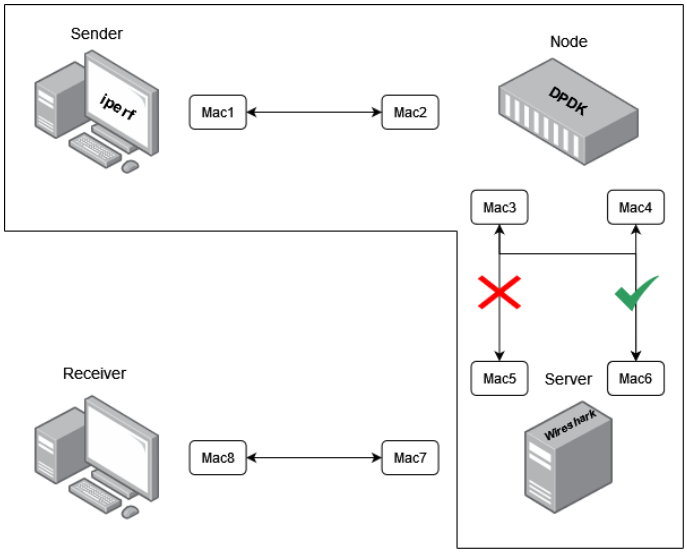

Ilustración 6: Caída Enlace

7.3 Diseño e implementación

7.3.1 Visión general

A continuación, se explicarán en detalle el software utilizado para la arquitectura definida anteriormente, centrándose en concreto en el nodo. Para el desarrollo de dicha solución, se ha utilizado el lenguaje de programación C aprovechando su rapidez junto con las librerías correspondientes a la tecnología DPDK la cual facilita la gestión de paquetes.

En esta implementación, para abordar tanto el encaminamiento de nivel 2 como la detección de la caída de enlaces, se ha tomado como referencia el código de ejemplo proporcionado por los desarrolladores de DPDK, sobre el cual se han realizado pequeñas modificaciones con el fin de adaptarlos a la solución requerida. Además, se ha realizado el desarrollo del código correspondiente para realizar la atención ante una interrupción producida por la caída del enlace y a su vez regresar a su estado original una vez que se recupere dicho enlace.

7.3.2 Diseño DPDK

A continuación, se especifica de forma más precisa el diseño del algoritmo utilizado en la solución tanto para la obtención y procesado de paquetes, como el funcionamiento del cambio de enlace.

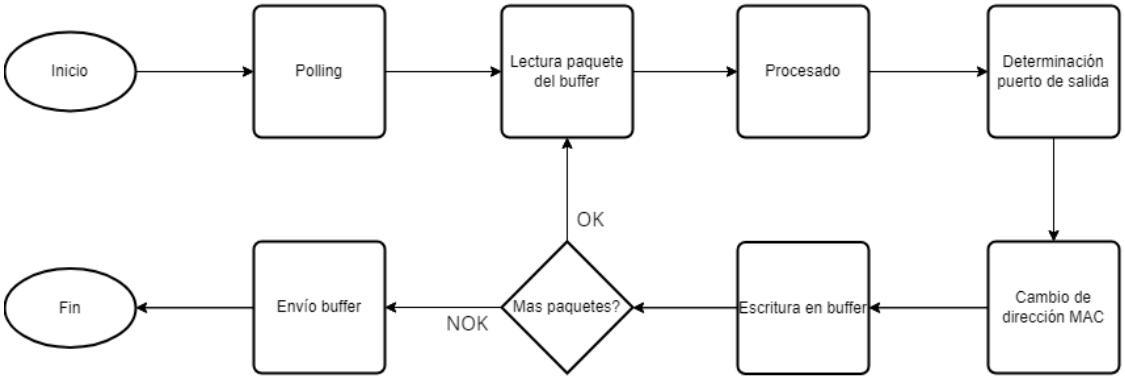

Ilustración 7: Diagrama obtención y emisión paquetes

Como ya se ha mencionado en la contextualización de este proyecto, DPDK hace uso del Poll Mode Driver para la obtención periódica de paquetes desde la interfaz de red a la memoria reservada para la aplicación. Para un funcionamiento optimo, se dedica un núcleo del procesador para cada puerto utilizado.

Esto permite evitar que un único núcleo realice búsqueda de paquetes puerto por puerto y principalmente garantiza que a pesar de que los puertos estén enviando paquetes, el programa será capaz de obtener los paquetes entrantes de forma simultánea.

El proceso comenzaría con dicho "Polling", donde se obtendrán un numero variable de paquetes dependiendo de las características del tráfico en ese momento. A continuación, se realiza la lectura del primer paquete obtenido y se procesa. Este procesado consiste principalmente en la comprobación del Checksum, y lectura de las direcciones tanto origen como destino del paquete. Una vez obtenida la dirección destino, se determina el puerto de salida del paquete siguiendo una tabla de forwarding.

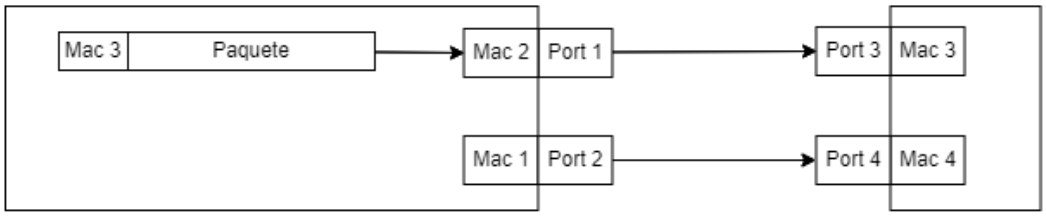

Ilustración 8: Forwarding caso 1

El cambio de la dirección MAC destino del paquete que se realiza a continuación únicamente ocurre en caso de que el puerto de salida determinado en el paso anterior corresponda a un puerto alternativo. En la Ilustración 8, podemos apreciar que el paquete saliente por el puerto 1 se le ha colocado la dirección destino Mac 3.

Sin embargo, a continuación, se muestra la situación en la que, por motivos de la caída del primer enlace, el paquete deberá tomar el puerto 2 con su correspondiente dirección de destino

Mac 4. Observando la Ilustración 7, se podría pensar que sería más optimo realizar el cambio de dirección Mac durante el procesado y más tarde determinar el puerto de salida.

Pero, como se explicará a continuación, la determinación del puerto de salida y el cambio de la dirección de destino pueden ocurrir en cualquier momento por medio de una interrupción. En este caso, se ha considerado que, con el fin de garantizar el menor número de paquetes descartados, el destino de cada paquete se decidide en los últimos pasos.

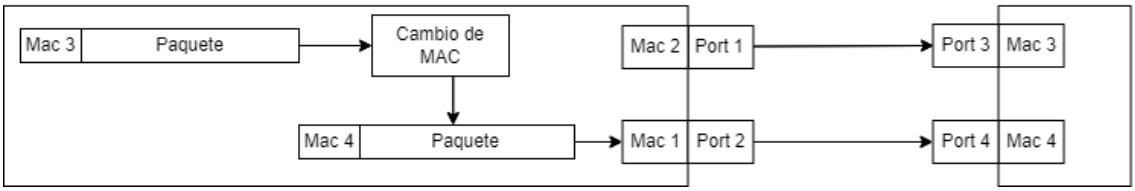

Ilustración 9: Forwarding caso 2

Durante la habilitación de los puertos disponibles, se registrará una función que atenderá las interrupciones producidas durante la ejecución del programa. Es decir, en paralelo a la recepción, procesado y envío de paquetes, se ejecutará un bucle encargado de comprobar y notificar el estado de los puertos.

En caso de que dicho bucle detecte la caída de un enlace, lo notificará por pantalla y modificara la entrada correspondiente a la tabla de encaminamiento. A pesar de que el puerto perteneciente al enlace caído permanezca inactivo, el bucle permanecerá comprobando el estado de todos los puertos. Esto se debe a la decisión de que en caso de que se recupere el puerto original, el flujo se reenviara por el mismo, por lo que es necesario seguir comprando tanto el estado del enlace caído, como el del resto de puertos para notificarlo en caso de que suceda.

Una vez que se produzca la interrupción correspondiente a la recuperación del enlace original, se reactiva el puerto de salida para que el flujo pueda retomar su enlace predeterminado. Durante todo el proceso anterior se tendrá en cuenta el ir actualizando la variable que almacena el número de puertos disponibles, ya que esto también evita que ciertos procesos cíclicos eviten realizar iteraciones innecesarias.

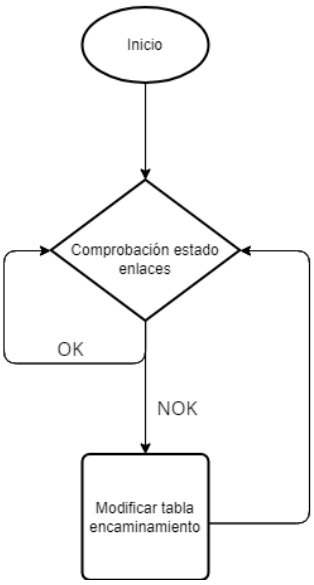

Ilustración 10: Atención a caídas de enlace

Como se ha podido apreciar, a lo largo del diseño se han tomado varias decisiones respecto a la reducción del tiempo de ejecución del programa y su optimización. Esto es esencial ya que considerando que trabajamos con grandes velocidades de transmisión de datos, cualquier pequeño intervalo que pueda evitarse durante la ejecución, garantizará el mejor rendimiento disponible.

7.3.3 Implementación

A lo largo de este apartado se describe el desarrollo de la solución diseñada mediante la descripción de sus características tanto globales como modulares.

7.3.3.1 Arquitectura de desarrollo

La arquitectura diseñada para el proyecto se centraría en facilitar el envío y recepción de paquetes para su análisis al llegar al servidor. Dicha arquitectura estaría compuesta de 3 equipos siendo uno de ellos el emisor de paquetes, un nodo intermedio y finalmente el receptor.

1. Equipo Emisor:

   - Función: Equipo encargado de realizar el envío de paquetes a frecuencia y tamaños diferentes.
   - Software: Para la realización de las pruebas se usarán aplicaciones como *"iperf"* o *"NetScanTools"* para establecer la configuración de los paquetes a enviar.
   - O.S. : Windows 10
   - Hardware: CPU ≫ i5-1135G7 | RAM ≫ 16 GB
   - Configuración de red: MAC ≫ 22:22:00:00:00:02

2. Nodo

   - Función: Este nodo se ha configurado con DPDK para la redirección de paquetes por un puerto primario. En caso de caída del enlace sería el encargado de redirigir el tráfico por el puerto secundario.
   - Software: Dado que se trata de un nodo intermedio, únicamente dispondrá de DPDK.
   - O.S. : Ubuntu 23.10.1
   - Hardware: CPU ≫ i7-4770 | RAM ≫ 16GB + 4GB Swp
   - Configuración de red: MAC ≫
     - MAC2: A0:36:9F:2D:EA:43
     - MAC3: A0:36:9F:2D:EA:40
     - MAC4: A0:36:9F:2D:EA:42

3. Servidor (Equipo receptor)

   - Función: Su funcionalidad principal es el obtener los paquetes para enviarlos al equipo final correspondiente. Para la realización de las pruebas, será el equipo en el que se capturaran los paquetes para su análisis.
   - Software: Wireshark 4.0.8 [16]
   - O.S. : Ubuntu 23.10.1 [18]
   - Hardware: CPU ≫ i3-2120 | RAM ≫ 12 GB + 4GB Swp
   - Configuración de red: MAC ≫
     - MAC5: A0:36:9F:26:0D:21
     - MAC6: A0:36:9F:26:0D:23

7.3.3.2 Implementación de la solución

En este apartado se profundizará en cada una de las características del código implementado. Principalmente se describirán las funcionalidades más relevantes y creadas con el fin de cumplir los objetivos establecidos para esta solución.

La Ilustración 11 representa como el main_loop() se inicializa en cada uno de los núcleos según el número de puertos utilizados y en caso de que se finalice la ejecución del programa se despertara el proceso encargado de la función main() y se realizara la limpieza de los parámetros correspondientes.

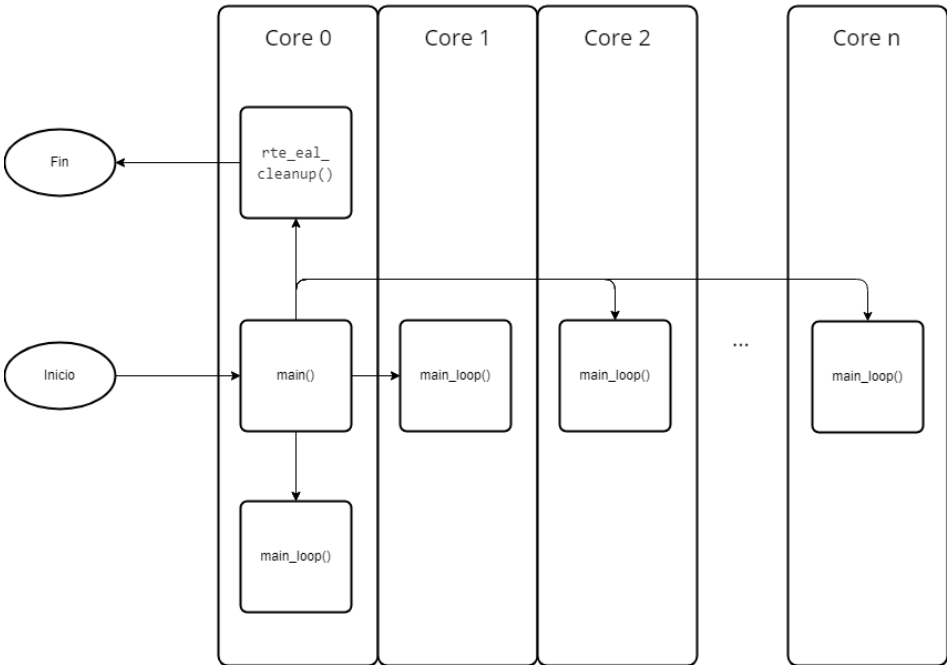

Ilustración 11: Esquema general

En función del número de puertos implicados en la maqueta, se utiliza un número determinado de núcleos del procesador. Como se puede observar en la Ilustración 11, la función principal *main(),* Comenzará ejecutándose en el núcleo 0, más tarde se profundizará más en sus características, pero de forma simple, esta función se encarga de la inicialización del hardware y de las llamadas a las funciones para cada puerto. Esta función denominada *main_loop(),* es la encargada de realizar todos los procesos que ocurren en cada puerto, se ejecuta una en cade núcleo por cada puerto implicado.

Es esencial que una vez que todos los bucles principales estén creados, el proceso *main()* pase a modo espera, esto permitirá que no obstaculice el funcionamiento del bucle situado en el núcleo 0.

Por último, una vez que se determine la finalización del programa, se ejecutara una pequeña función para la limpieza de ciertas configuraciones y finalizara su ejecución.

7.3.3.2.1 Inicialización

Como se ha mencionado anteriormente, el proceso de inicialización se producirá en la función *main()*. A continuación, se incluirán las configuraciones más relevantes:

1. Inicialización parámetros EAL:

   - Los parámetros EAL (Environment Abstraction Layer), son los que permiten aplicar la configuración necesaria para acceder al hardware del equipo. Toda aplicación al ser ejecutada requiere que se introduzca estos parámetros teniendo en cuenta las limitaciones físicas hardware y las necesidades del programa a ejecutar. Un ejemplo sería el siguiente:

     ./dpdk-app -l 0-3 -n 2 -- -p 0x7

   - En esta ejecución, los parámetros EAL son los situados entre la llamada del programa y el doble guion "--". Entre una gran variedad de parámetros, los más importantes son "-l" para indicar el número de núcleos dedicados exclusivamente a esta aplicación, y -n para indicar el número de tarjetas de memoria RAM disponibles.

2. Inicialización de los puertos:

   - En este paso se comienza creando y configurando las colas de paquetes en cada núcleo. Estas colas se almacenan una zona previamente reservada de memoria. Para ello se usa el concepto de hojas grandes "hugepages".

     El tamaño normal de las páginas en memoria suele ser entre 2KB y 8KB, considerando las de 2MB como paginas extremadamente grandes usadas para casos muy concretos. Sin embargo, para este tipo de aplicaciones, se recomienda el uso de páginas de 1GB. Esto se debe a que reduce la actualización del estado de las páginas ya que en vez de disponer de las tablas de páginas y TLB (Translation Lookaside Buffer) llenas de entradas, se reduce a una cantidad mucho menor. Además, esta característica reduce la fragmentación en memoria, se realiza un aprovechamiento más eficaz ya que permite al programa situar dinámicamente sus espacios de memoria en cada hoja.

   - Una vez de tener disponibles los núcleos y colas de mensajes, se procede a la inicialización de los puertos. A lo largo de este proceso, se obtiene información como el número de puertos disponibles o su estado actual. Entre la configuración más relevante podemos destacar el establecimiento de la MTU (Maximum Transmission Unit) permitida para cada puerto y la obtención de las direcciones MAC de cada interfaz.

```
rte_eth_dev_set_mtu(portid,mymtu);
ret = rte_eth_macaddr_get(portid,&lsi_ports_eth_addr[portid]);
```

Ilustración 12: Configuración Puertos

Junto con la adjudicación de tanto una cola de recepción como la de transmisión por núcleo con cada uno de los puertos. Se ha añadido una función para extraer de un archivo de configuración las direcciones MAC de los puertos situados al otro lado de cada enlace. Esto facilitara la función de reenvío de paquetes.

```
int get_dst_mac(uint16_t portnum){

    char mac_str[18];
    int i=0;
    FILE *fp = fopen("path","r");

    if(fp == NULL){
        perror("erorr al abrir el archivo.");
        return 1;
    }

    i=0;
    while(fgets(mac_str, sizeof(mac_str),fp) != NULL || i<portnum){

        mac_str[strcspn(mac_str,"\n")]='\0';
        if(mac_str[0]=='\0'){
            continue;
        }
        if ((lsi_enabled_port_mask & (1 << i)) == 0) {
            i++;
            continue;
        }

        if(rte_ether_unformat_addr(mac_str,&dst_info[i].mac) < 0 || rte_ether_unformat_addr(mac_str,&dst_table[i].mac) < 0){
            printf("Error con %s.",mac_str);
            fprintf(stderr, "Error al convertir la direccion.");
            return 1;
        }
        dst_info[i].port=i;
        dst_table[i].port=i;
        lsi_dst_backup[i]=i+1;
        printf("Dest Port %d , MAC address: " RTE_ETHER_ADDR_PRT_FMT "\n",dst_info[i].port,RTE_ETHER_ADDR_BYTES(&dst_info[i].mac));
        i++;
    }
    fclose(fp);

    return 0;
}
```

Ilustración 13: Función configuración MAC destino

Como paso previo a la detección de la caída de los enlaces, es esencial el realizar un registro de la función que atenderá a dichas caídas para cada uno de los puertos. En posteriores apartados se explica en más profundidad.

Por último, se realizará una comprobación del estado de cada puerto disponible. Esto sirve para inicializar las variables que almacenan las tablas de forwarding y su configuración correspondiente

```
check_all_ports_link_status(nb_ports, lsi_enabled_port_mask);
```

Ilustración 14: Comparación estado de enlace

7.3.3.2.2 Variables de forwarding

Antes de proceder con la profundización en el funcionamiento de los eventos de forwarding y caída de enlace, mencionaremos las variables más relevantes para el manejo de esas funcionalidades.

```
struct fwd_dst_info {
    struct rte_ether_addr mac;
    unsigned port;
} __rte_cache_aligned;

struct fwd_dst_info dst_info[RTE_MAX_ETHPORTS];

struct fwd_dst_info dst_table[RTE_MAX_ETHPORTS];

static unsigned lsi_temp_port[RTE_MAX_ETHPORTS] = {0} ;
static unsigned lsi_changed_port[RTE_MAX_ETHPORTS] = {0};
```

Ilustración 15: Variables direccionamiento

- **dst_info**: Almacena las direcciones MAC destino junto con sus correspondientes puertos de salida originales. Este conjunto de estructuras no se modifica salvo en caso de reconfiguración de la arquitectura.

- **dst_table**: Tabla de forwarding, que almacena la dirección destino junto al puerto de salida. Esta tabla varía según el estado de los enlaces, es decir, solo aparecerán los puertos que están activos y disponibles para su uso.

- **lsi_temp_port**: Variable de almacenamiento temporal. Se utiliza principalmente para garantizar un intercambio seguro de los puertos en la tabla de forwarding.

- **lsi_changed_port**: En esta variable se guarda el estado para cada dirección MAC destino, donde 0 indica que la MAC destino en dicha posición tiene en su tabla de forwarding su puerto predeterminado, y 1 indica que se ha editado el puerto de salida en la tabla con su puerto alternativo por la caída de un enlace.

7.3.3.2.3 Basic Forwarding

Con un funcionamiento básico, se lleva a cabo el siguiente procedimiento para la obtención, procesado y envío de paquetes.

1. El paquete obtenido de la cola de entrada se procesa comprobando el Checksum y otros campos.

2. Se realiza la búsqueda de la dirección Mac destino del paquete y se obtiene su puerto correspondiente de salida.

3. Por último, por motivos de optimización, DPDK añade bytes a los paquetes de un tamaño menor a 60 bytes hasta llegar a dicho tamaño, esto lo realiza justo antes del envío. A continuación, se direcciona el paquete a su puerto de salida.

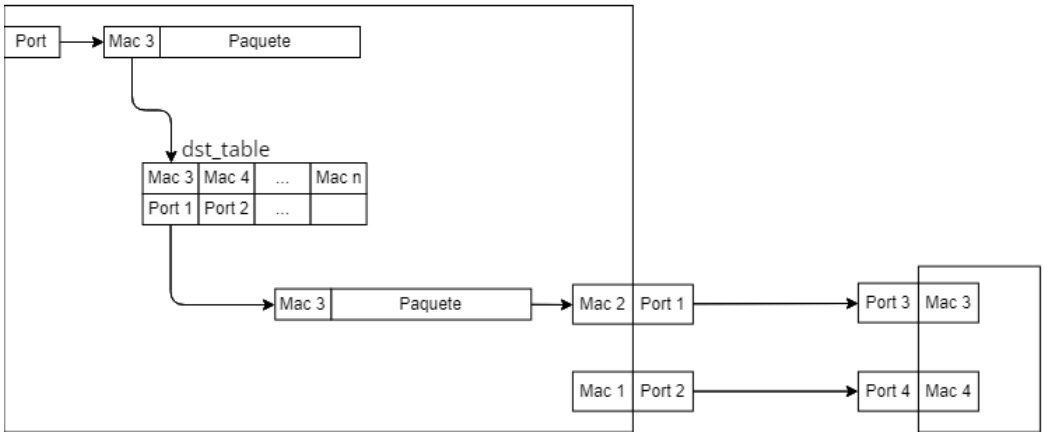

Ilustración 16: Basic Forwarding

7.3.3.2.4 Detección caída del enlace

Durante su funcionamiento puede que se produzca la caída del enlace bien por un factor de la red en sí, o incluso por un factor externo como obras o desgaste del cableado. Por ello se hacen uso de conexión redundantes, para redirigir el tráfico por su nuevo enlace y así tener tiempo suficiente para organizar y realizar el mantenimiento correspondiente.

```
rte_eth_dev_callback_register(portid,
    RTE_ETH_EVENT_INTR_LSC, lsi_event_callback, NULL);
```

Ilustración 17:Registro función Callback

Como se ha mencionado anteriormente, durante la etapa de inicialización se incluye una función la cual habilitara la llamada a una función concreta cada vez que se produzca la interrupción por el cambio de estado de un puerto.

Mediante el parámetro *RTE_ETH_EVENT_INTR_LSC,* es como se indica este tipo de eventos para un puerto con un id *portid.* La función que se ejecuta es *lsi_event_callback()* y es importante tener en cuenta que este tipo de atención a interrupciones las ejecuta el núcleo en el que se ha sufrido dicha interrupción, esto es esencial ya que permite que el resto de los puertos no sufra ni un mínimo retardo a la hora de continuar con su actividad.

7.3.3.2.5 Respuesta ante la caída

A continuación, se mencionan los pasos que se realizan desde que un puerto sufre una interrupción.

1. Se genera la interrupción, por lo que se realiza una llamada a la función *"lsi_event_callback*()" la cual tiene a la siguiente estructura:

```
ret = rte_eth_link_get_nowait(port_id, &link);

if(link.link_status){
    //"Link up event"
    lsi_event_up();
}else{
    //"Link down event"
    lsi_event_down();
}
```

Ilustración 18: Estructura función event_callback

Donde, port_id es un parámetro obtenido por parte de la interrupción.

2. En el caso de la caída, se guarda en la tabla *lsi_temp_port* el valor de los puertos originales de forma temporal para en caso de que se recupere el enlace, se devuelva el estado original a la tabla de forwarding.

3. A continuación, se modificará la tabla de forwarding *dst_table* con el puerto alternativo. Para esta solución, se ha supuesto que el puerto alternativo será el propio incrementado en una unidad menos para el puerto n, el cual tendrá como puerto alternativo el 0. Sin embargo, se pueden aplicar cualquier tipo de política respecto a la decisión del puerto alternativo como seleccionar el enlace más rápido o el menos usado entre otras configuraciones.

4. Una vez modificado el puerto destino, se indica en la tabla *lsi_changed_port* con 1 las direcciones que han sido modificadas. Esto ayudara posteriormente a identificar la disponibilidad de los enlaces.

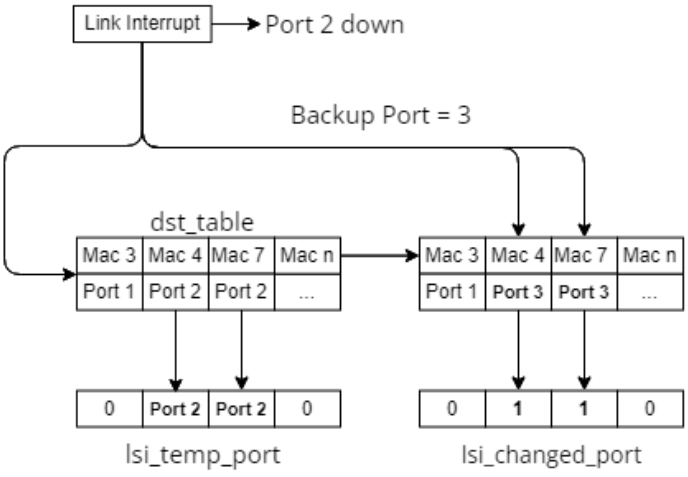

Ilustración 19: Atención a la interrupción

7.3.3.2.6 Funcionamiento alternativo

Con el nuevo enlace alternativo, el procedimiento para direccionar los paquetes es ligeramente diferente, ya que a pesar de que el equipo origen disponga de la dirección MAC correcta de destino, este no es consciente de que dicho enlace ya no está disponible. Es decir, es responsabilidad del nodo de que, a pesar de dicha caída, tanto para el quipo origen como destino el evento sea lo más imperceptible posible.

1. Se obtiene el paquete de la cola de entrada y se hace la lectura de su dirección MAC destino junto con su procesado correspondiente.

2. Según la tabla de forwarding modificada por la caída del enlace, se le adjudica el puerto destino a dicho paquete.

3. Como el puerto destino se trata de un puerto alternativo por la caída de un enlace, se deberá hacer uso de la tabla *dst_info* para obtener la dirección MAC real del equipo en dicho puerto alternativo.

4. Por último, se envía el paquete por el puerto alternativo con la dirección MAC destino correspondiente.

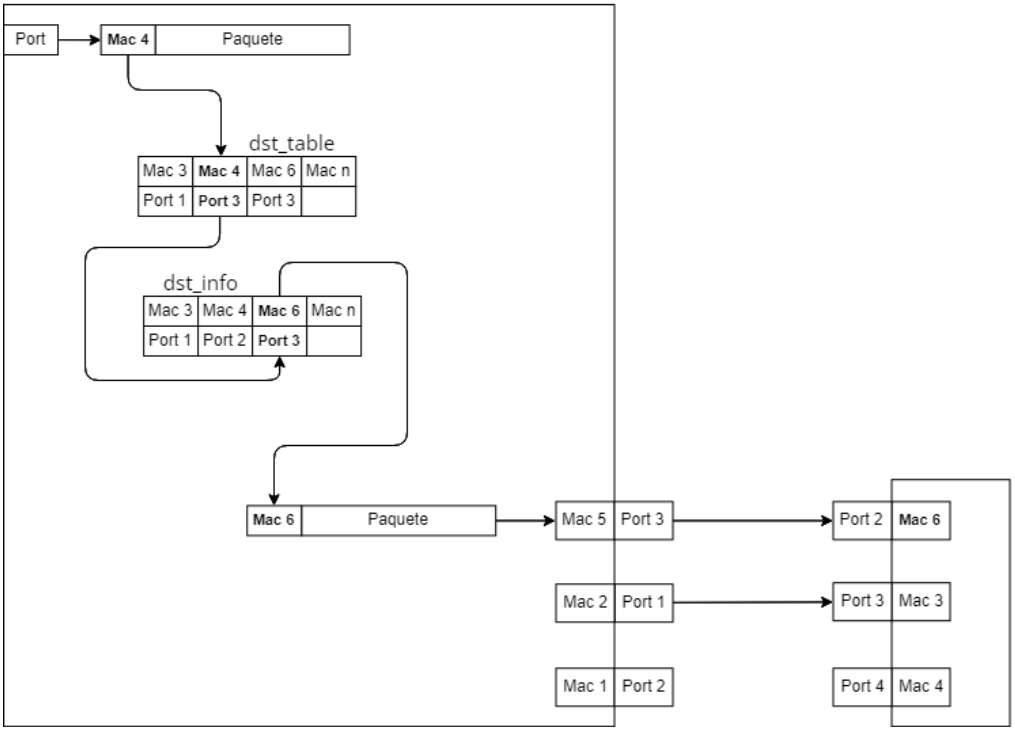

Ilustración 20: Backup Forwarding

7.3.3.2.7 Recuperación del enlace

Para esta solución, según se produzca un evento de recuperación de un enlace, se retoma el uso de este con el siguiente procedimiento.

1. Se realiza la llamada a la función de atención a interrupciones indicando el identificador del puerto.

2. Recuperando de la tabla temporal los puertos originales se sustituyen en la tabla de forwarding.

3. Por último, ya que en ese momento se está usando un puerto predeterminado, se colocan los flags de la tabla *lsi_changed_port* a 0.

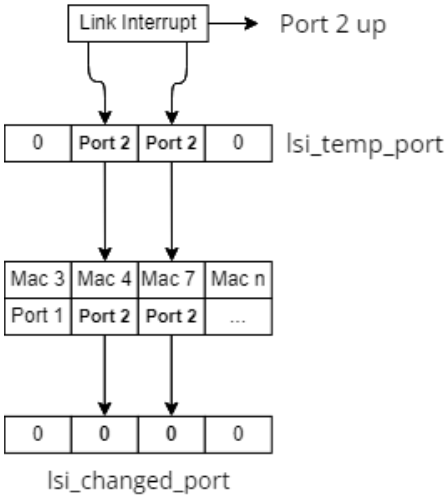

Ilustración 21: Atención a interrupción up

7.4 Pruebas de evaluación y análisis de rendimiento

En esta sección se incluirán todos los detalles relativos a las pruebas realizadas del proyecto. Se comienza desarrollando la maqueta utilizada junto con su correspondiente plan de pruebas terminando con la presentación de los resultados obtenidos en las mismas.

7.4.1 Diseño plan de pruebas

A continuación, se realiza una descripción del plan de pruebas junto con cada prueba que se ha realizado individualmente.

Independientemente del tipo de prueba que se realiza, se ha tenido en cuenta una serie de condiciones a cumplirá lo largo de las mismas. Esto se lleva a cabo principalmente para evitar la obtención de valores extremos que no representan el funcionamiento real de la solución.

Plan de pruebas:

1. Generación de paquetes constante, evitando variaciones respecto al ancho de banda establecido de no más de 5/10 Mbps, con ello se evita influir en pruebas como tiempo de caída.

2. Realizar desconexión aproximadamente en la mitad del periodo de duración de cada prueba esto es debido a que la caída del enlace forzada podría comportarse de diferentes formas si se realiza con demasiada cercanía al inicio de la prueba.

3. La desconexión o caída de un enlace se realiza manualmente para evitar cualquier retardo que se podría producir al hacerlo por software, garantizando una desconexión correcta y principalmente que dicha desconexión se produzca de la manera más realista posible.

4. La medida denominada tiempo de recuperación será medida como la diferencia de tiempos entre el primer paquete recibido por el puerto alternativo y el último paquete recibido por el puerto principal.

5. La pérdida de paquetes será medida con la diferencia de los paquetes enviados (valor proporcionado por la herramienta iperf y wireshark en el equipo origen) y los recibidos en el equipo destino (medido con wireshark)

6. A pesar de que la tecnología DPDK soporte los paquetes denominados “Jumbo Frames” (mayores que 1500 bytes), a lo largo de las pruebas se utilizaran únicamente de un tamaño de 1500 bytes.

7. Para la medida de cada valor, se ha estimado realizar 10 pruebas, con las cuales se realizará el cálculo promedio.

8. Respecto a la captura de los paquetes, estas se realizan mediante wireshark tanto en origen como el destino de las pruebas.

9. En caso de producirse cualquier alteración bien por rendimiento u otro motivo por parte del generador de paquetes, se repetirá dicha prueba. Lo mismo ocurrirá si la desconexión se produce con excesiva cercanía al inicio de la prueba.

7.4.2 Prueba de rendimiento básico

Mediante la herramienta iperf se envían paquetes a una velocidad durante un periodo de tiempo. Para cada prueba se incrementará la velocidad con el fin de estimar si los enlaces efectivamente soportan la configuración establecida de 1Gbps. A continuación, se detalla con más profundidad dichas pruebas.

Tabla 5: Plan de Pruebas básicas

Nº de prueba	Ancho de banda (Mbps)	Duración	Streams paralelos
00	5	30	1
01	100	30	1
02	125	30	1
03	200	20	1
04	500	20	1
05	600	15	1
06	850	10	1
07	1000	10	1
08	500	10	2
08	500	10	3

Se ha considerado reducir la duración de la prueba a medida que el ancho de banda incrementa debido el excesivo tamaño de las capturas realizadas y almacenadas para su posterior análisis.

7.4.3 Análisis caída y recuperación del enlace con DPDK

Durante esta prueba se analizará específicamente el rendimiento de la solución a la hora de lidiar con la caída de un enlace. Para ello, se han realizado 10 pruebas consecutivas para cada uno de los anchos de banda incluidos a continuación.

Tabla 6: Prueba Caída del enlace DPDK

Nº de prueba	Ancho de banda (Mbps)	Tiempo de la prueba
00	100	10
01	300	10
02	500	10
03	750	10

Para dichas pruebas se analizarán los siguientes parámetros:

Tiempo de recuperación: Tiempo en el que DPDK deshabilita el enlace caído y reanuda el flujo por el enlace alternativo.

Perdida de paquetes: El número de paquetes perdidos debido a la desconexión. Este valor varía según el ancho de banda y el rendimiento del nodo en el momento de la prueba.

7.4.4 Análisis caída y recuperación del enlace con tablas de direccionamiento predeterminadas

Esta prueba pretende realizarse con el fin de ser comparada con la realizada mediante DPDK. Para ello se ha realizado una reconfiguración de nuevo usando exactamente los mismos equipos con el mismo software haciendo uso de las tablas de direccionamiento predeterminado del sistema Linux.

Para ello únicamente se ha requerido habilitar el nodo para que realice direccionamiento entre dos equipos, ya que esto no es una característica por defecto.

Las pruebas se realizan con los siguientes escenarios:

Tabla 7: Prueba Caída del enlace sin DPDK

Nº de prueba	Ancho de banda (Mbps)	Tiempo de la prueba
00	100	10
01	300	10
02	500	10
03	750	10

Al igual que en las pruebas con DPDK, se analizarán los siguientes parámetros:

- Tiempo de recuperación
- Perdida de paquetes

7.4.5 Resultado de las pruebas

A lo largo de este apartado, se presentan los resultados del plan de pruebas anteriormente desarrollado y se analizan en profundidad con el fin de comprobar su rendimiento y su comparación con un reenvío básico.

7.4.5.1 Prueba de rendimiento básico

Durante las pruebas de funcionamiento básico se ha podido apreciar que DPDK ha sido capaz de mantener el flujo de forma constante de los 5Mbps a los 850/900Mbps. A continuación, podemos apreciar la captura de los paquetes UDP obtenidos en el equipo de destino.

Max: 5,17 Mbps

Min: 5,06 Mbps

Promedio: 5,14 Mbps

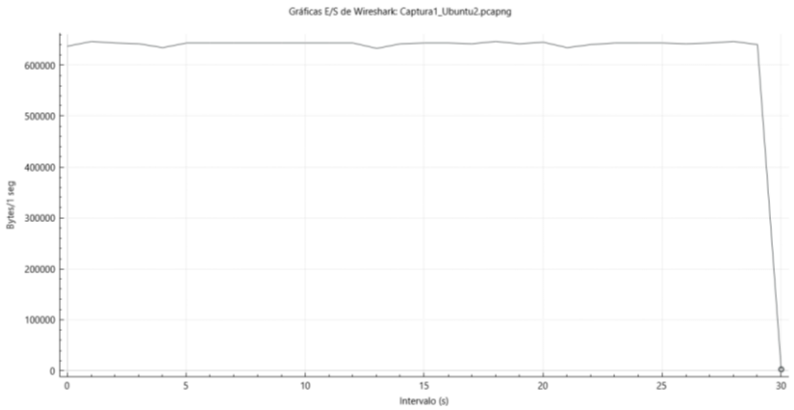

Ilustración 22: Grafica E/S en destino a 5Mbps

En la ilustración atención se puede ver como se producen pequeñas variaciones, pero se mantiene en los 5Mbps correctamente.

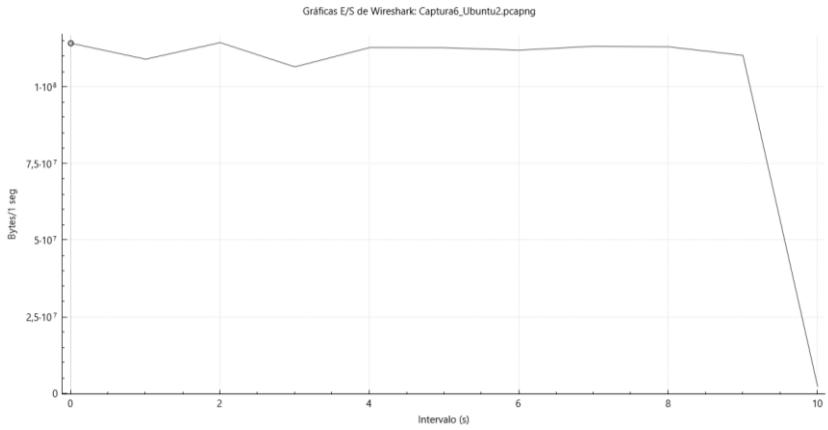

Ilustración 23: Grafica E/S en destino 900Mbps

Por otro lado, en la ilustración 23 podemos apreciar que se ha producido una pequeña variación, a pesar de ello, se ha mantenido relativamente constante con los siguientes valores:

Max: 916,3 Mbps

Min: 853 Mbps

Promedio: 895 Mbps

Como un caso extremo, la última prueba consiste en el envío 3 flujos a 500 Mbps cada uno. Teniendo en cuenta que se trata de un enlace de 1Gbps, el ancho de banda para cada uno de los flujos de maneara teórica correspondería a 333 Mbps.

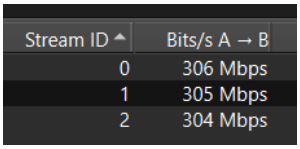

Ilustración 24: Ancho de banda 3 flujos a 500 Mbps

De forma satisfactoria se ha podido probar que a pesar de que en la práctica no llegue a dichos valores extremos, se acerca considerablemente a los 306 Mbps por stream con un ancho de banda promedio de todos ellos de 916 Mbps.

Tabla 8: Tabla 3 flujos

(Mbps)	Stream 0	Stream 1	Stream 2	Total
Max	327	331	326	984
Min	305	299	296	892
Promedio	306	305	304	916

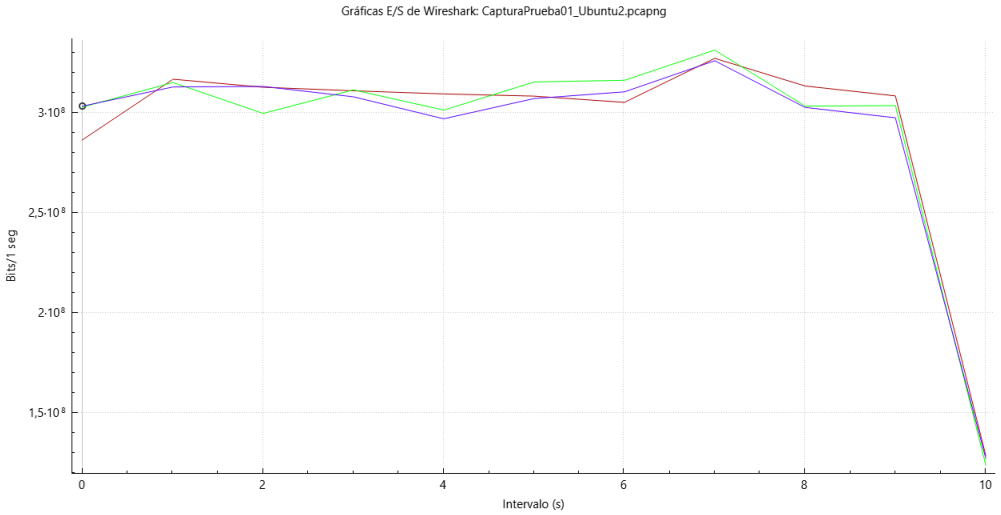

Ilustración 25: Grafica E/S en destino 3x500Mbps

7.4.5.2 *Análisis caída y recuperación del enlace con DPDK*

Para estas pruebas como se ha mencionado en el plan de pruebas, se realiza una desconexión manualmente del enlace predeterminado mientras se realiza el envío de paquetes. Se espera que DPDK reaccione a dicha caída lo más rápido posible y reanude el tráfico de nuevo.

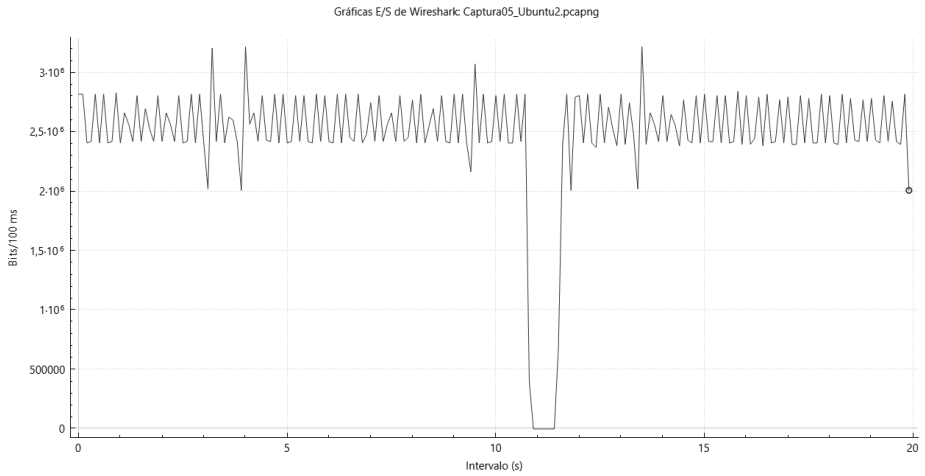

Ilustración 26: Caída del enlace a 25Mpbs

En este ejemplo se puede observar cómo se produce la caída del enlace para un flujo de 25 Mbps. Se observa cómo a pesar de producirse la caída entre el segundo 10 y el 11, se recupera correctamente y sobre todo el flujo de paquetes parece proseguir con la misma constancia que en momentos previos a la desconexión.

```
10.814588221   22:22:00:00:00:00
10.814760675   22:22:00:00:00:00
11.579188067   22:22:00:00:00:01
11.579371716   22:22:00:00:00:01
```

Ilustración 27: Tiempo de recuperación

Para determinar cuál es el tiempo que ha necesitado DPDK para deshabilitar el enlace caído y hacer uso del enlace nuevo, se ha identificado los paquetes ultimo antes del corte y el primero después del corte mediante su dirección MAC destino. Con esos dos valores de tiempo se restan y se obtiene dicho valor.

$$11{,}579188067 - 10{,}814760675 = 0{,}7644274 \text{ s}$$

En este caso concreto se ha obtenido un valor de 0,7644 s con una pérdida de 1.662 paquetes de un total de 42534 paquete transmitidos durante toda la prueba. Lo que correspondería a un 4% de paquetes perdidos en 20 segundos de prueba.

Tomando este caso como ejemplo, se han realizado 10 pruebas de una duración de 10 segundos para diferentes anchos de banda calculando el tiempo de recuperación, los paquetes perdidos, y cuanto seria la proporción de paquetes perdidos respecto a los totales. A continuación, se muestra un ejemplo para un ancho de banda de 100 Mbps.

	Prueba 100 Mbps							
			Tiempo	Paq Enviados	Recibidos	Perdidos	%Perdidos	Paq perdidos por segundo
01	5,996285642	5,241107397	0,755178245	85.355	81.935	3.420	4,007	4.528,732154424
02	5,865718627	5,113727867	0,751990760	85.344	82.146	3.198	3,747	4.252,711828534
03	5,988289333	5,236342311	0,751947022	85.348	82.119	3.229	3,783	4.294,185501808
04	6,382103575	5,630006692	0,752096883	85.360	82.149	3.211	3,762	4.269,396766001
05	6,059662924	5,309246056	0,750416868	85.349	82.255	3.094	3,625	4.123,041647832
06	6,207780378	5,456262587	0,751517791	85.354	82.139	3.215	3,767	4.278,009168249
07	6,310393107	5,558574719	0,751818388	85.350	82.117	3.233	3,788	4.300,240658652
08	6,565184553	5,814444837	0,750739717	85.348	82.183	3.165	3,708	4.215,841962852
09	6,286485696	5,534656203	0,751829493	85.377	82.185	3.192	3,739	4.245,643499915
10	6,861145764	6,110780988	0,750364776	85.346	82.147	3.199	3,748	4.263,259820181
		Promedio	0,751789994	85.353	82.138	3.216	3,767	

Ilustración 28: 10 Pruebas para 100Mbps

Se puede apreciar como para las 10 pruebas el tiempo de recuperación en promedio es de 0,75 segundos, en tiempo razonable en el que únicamente se produce la pérdida del 3,76% de los paquetes enviados.

Tabla 9: Resultados pruebas DPDK

Ancho de banda (Mbps)	Tiempo de recuperación (s)	% Paquetes perdidos

100	0,7517	3,767
300	0,7525	3,446
500	0,7507	7,702
750	0,7509	7,898

En la tabla superior se puede observar como el tiempo de recuperación para los 4 anchos de banda estudiados está situado alrededor de los 0,75 s, sin embargo, al alcanzar los 500 Mbps dicho tiempo de recuperación sufre una penalización mayor respecto al porcentaje de paquetes perdidos. Esto puede deberse en parte por el rendimiento del hardware utilizado en la maqueta de pruebas y por la dificultad de DPDK de reponerse a dicha caída al intentar mantener grandes velocidades de transmisión.

7.4.5.3 Análisis caída y recuperación del enlace con tablas de direccionamiento predeterminadas

Una vez obtenidos los catos correspondientes a la solución diseñada, se procede a realizar una comparación con el fin de confirmar la mejoría respecto a un direccionamiento haciendo uso de las tablas IP predeterminadas.

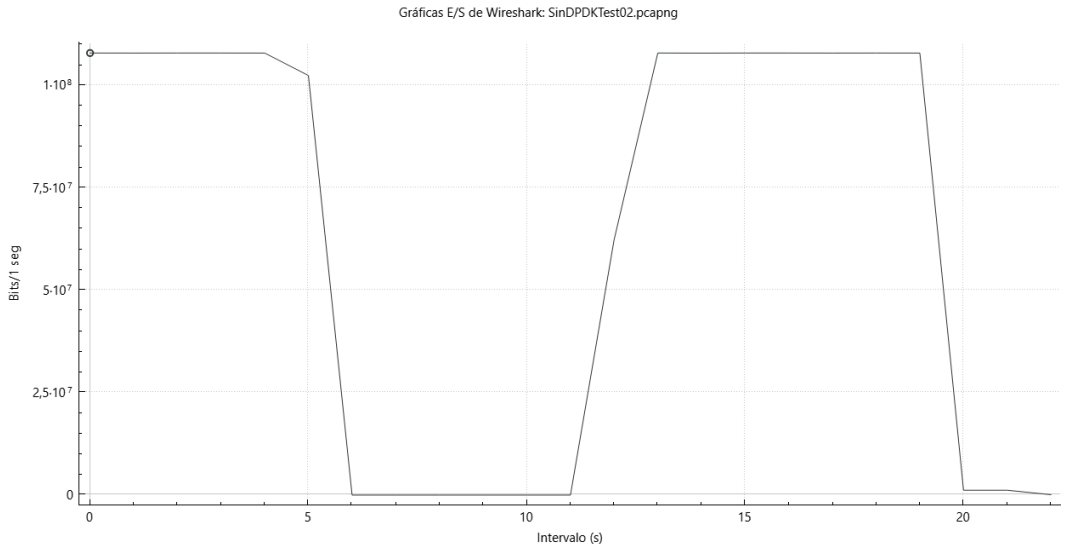

Ilustración 29: Prueba sin DPDK

En el ejemplo de la prueba superior, se ha utilizado un ancho de banda de 100 Mbps usando exactamente la misma configuración de paquetes con el mismo software, iperf, y con la misma configuración hardware en el nodo. En esta situación, se observa la necesidad de un tiempo de recuperación desde el segundo 6 al 12.

$$12{,}4223 - 5{,}9810 = 6{,}44 \text{ s}$$

```
5.981033722   IntelCor_2d:ea:40
5.981033768   IntelCor_2d:ea:40
12.422390077  IntelCor_2d:ea:42
12.422390365  IntelCor_2d:ea:42
```

Ilustración 30: Tiempo de recuperación sin DPDK

Para este ejemplo se ha calculado un tiempo de recuperación de 6,44 s con una pérdida de paquetes de un 20% de perdida de paquetea. Para todas las pruebas realizadas de un tiempo de 20 segundos debido al excesivo tiempo de recuperación el porcentaje de perdida de paquetes se encuentra entre el 18,46 y 22,923 por ciento.

7.4.6 Conclusión de los resultados

Considerando los resultados obtenidos, se puede afirmar que la utilización de las librerías DPDK puede considerarse como una opción muy interesante para una red de telecomunicaciones.

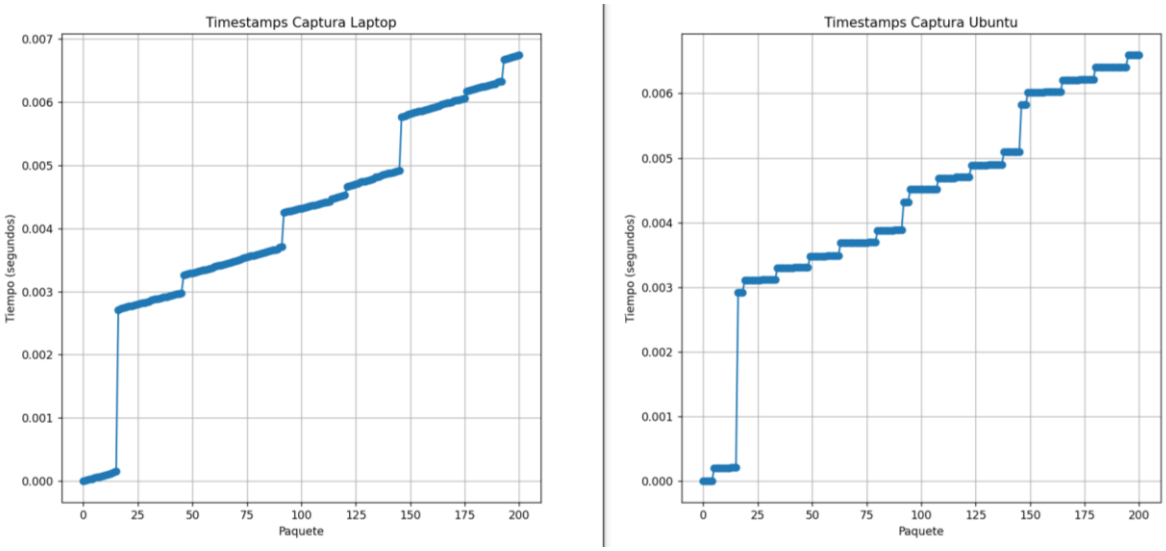

Ilustración 31: Flujo de entrada vs salida

Es importante mencionar que DPDK dispone de un funcionamiento peculiar mediante el cual envía paquetes agrupados. Esto en diversas aplicaciones podría tener cierto conflicto, sin embargo, se ha podido confirmar que tendría los siguientes beneficios:

- Batch Processing: Permite la optimización del uso de la CPU mediante el procesado de paquetes por grupos.

- Mejora throughput: A pesar de que pueda parecer una contradicción, el envío de paquetes en "bursts" permite obtener un throughput mayor con el pequeño sacrificio de una mayor latencia en ciertos paquetes.

Por último, hay que destacar la considerable mejora de la recuperación del enlace ante una caída. Haciendo uso de las tablas IP tradicionales se ha observado un tiempo de recuperación de más de 6 segundos, sin embargo, mediante DPDK únicamente con 750 milisegundos es suficiente para retomar el tráfico por un enlace alternativo.

8 Metodología

8.1 Descripción de fases

Fase 1: Definición de planificación, objetivos y presupuesto

Esta primera es en la que se dan los primeros pasos del proyecto especificando los objetivos a llevar a cabo y el presupuesto.

T1.1	**Definición objetivos**: Durante esta tarea, se definen los objetivos a cumplir durante el periodo del proyecto.
T1.2	**Definición presupuesta**: Como tarea complementaria a la anterior, se definirá el presupuesto disponible a ser utilizado.

Hito 1:

Con la finalización de la primera fase se documenta toda la información definida y se documentaran tanto los objetivos como el presupuesto.

Recursos: Microsoft Word

Duración total: 2 semanas

Fase 2: Diseño de solución

La segunda fase consta de tres tareas en las que se llevara a cabo el diseño de la solución mediante la cual se cumplirán los objetivos definidos de la primera fase.

T2.1	**Selecciones alternativas**: Se valoran todas las posibles alternativas a utilizar para el diseño de la solución.
T2.2	**Diseño infraestructura global:** Se define la solución de forma global, especificando que se quiere obtener y mediante que método.
T2.1	**Diseño componentes**: Se realiza el diseño de cada componente participante en la solución, teniendo extremada precaución en la definición de las interfaces que las unen.
T2.2	**Diseño de equipos**: Se define la configuración especialmente de los equipos que son utilizados en la infraestructura diseñada

Hito 2:

Para finalizar la segunda fase, se incluye un informe en el que se detallan las decisiones tomadas respecto a las alternativas valoradas y el diseño final de la solución tanto general como la definida por componentes.

Recursos: Documentación tecnologías, artículos

Duración total: 8 semanas

Fase 3: implementación de la solución

La fase de implementación consiste en llevar a cabo el diseño planteado en la fase anterior.

T3.1	**Instalaciones**: Se realiza la configuración e instalación del software necesario para realizar la correcta implementación de la solución.
T3.2	**Implementación modelo básico**: Esta tarea se centra en comenzar la implementación del modelo inicial con las funcionalidades más básicas de la solución.
T3.1	**Implementación modelo completo**: Una vez finalizado el modelo básico, se implementa sobre este todas las funcionalidades más complejas y personalizadas para la solución concreta.

Hito 3:

Se completa el manual de instalación del entrono utilizado con el fin de ser usado como respaldo ante cualquier incidente.

Hito 4:

Se documenta la implementación realizada a lo largo de la fase.

Recursos: Software programación, equipos de trabajo remoto y local

Duración total: 7 semanas

Fase 4: Diseño y despliegue plan de pruebas

Se realiza tanto el diseño como el despliegue de las pruebas necesarias para analizar el correcto funcionamiento de la implementación al mismo tiempo que se realiza un estudio del rendimiento de esta.

T4.1	**Diseño maqueta de pruebas**: Se definen los equipos e infraestructura a utilizar para la realización de las pruebas.

T4.2	**Diseño del plan de pruebas**: Se diseña todas las características de las pruebas a realizar de las que podemos destacar software a utilizar, parámetros, resultados esperados, etc.
T4.1	**Despliegue de pruebas**: Una vez finalizada el diseño de maqueta y plan de pruebas, se procede a su despliegue y puesta en marcha. En esta tarea se contemplan todas las situaciones planteadas en el plan de pruebas.
T4.2	**Análisis y valoración de resultados:** Con la información obtenida de la tarea tres, se realiza el estudio e interpretación de los datos obtenidos.

Hito 5:

Para la finalización de la fase 4, si incluirá un informe detallado de las pruebas realizadas y sus correspondientes resultados.

Recursos: Equipamiento en la maqueta, software de pruebas

Duración total: 6 semanas

Fase 5: Gestión y documentación

En la última fase del proyecto se procede a la redacción de toda la documentación recopilada a lo largo del mismo.

T5.1	**Puesta en marcha y seguimiento:** Se realiza la puesta en marcha del proyecto con su correspondiente seguimiento.
T5.2	**Redacción documentación:** Se unifica la documentación realizada y se incluyen la necesaria para la finalización del informe.

Hito 6:

Se realiza la entrega de la memoria del proyecto

Recursos: Microsoft Word

Duración total: 3 semanas

8.2 Cronograma

A continuación, se incluye el cronograma donde se incluyen las fases y tareas mencionadas anteriormente junto con su correspondiente duración.

El proyecto se ha realizado en una duración de 7 meses en sus correspondientes 5 fases.

Duración	213 días
Horas de trabajo	302

A continuación, es incluyen el listado de hitos a completar

Tabla 10: Hitos

Código	Semana	Titulo
H1	2	Documentación objetivos y presupuesto
H2	10	Finalización de diseño
H3	17	Manual de instalación
H4	17	Finalización de implementación
H5	23	Finalización de pruebas
H6	26	Entrega memoria

		MES1				MES2				MES3				MES4				MES5				MES6				MES7			
	Semana	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
FASE 1	**Definición de planificación, objetivos y presupuesto**	3	3																										
T1.1	**Planificación del proyecto**	2																											
T1.1.1	Definición objetivos	1																											
T1.1.2	Definición presupuesto	1																											
H1	**Documentación objetivos y presupuesto**		X																										
FASE 2	**Diseño solución**			3	3	3	3	3	3	3	3																		
T2.1	**Selección alternativas**			2																									
T2.2	**Diseño infraestructura global**				2																								
T2.3	**Diseño componentes**				2	2	2	2																					
T2.4	**Diseño equipos**								2	2																			
H2	**Fonalizacion diseño**										X																		
FASE 3	**Implementación de la solución**											3	3	3	3	3	3	3											
T3.1	**Instalación y configuración**											2	2																
T3.2	**Implementación modelo básico**													2	2														
T3.3	**Implementación modelo completo**															2	2												
H3	**Anexo manual de instalación**																	X											
H4	**Fonalizacion de implementación**																	X											
FASE 4	**Diseño y despliegue plan de pruebas**																		3	3	3	3	3	3					
T4.1	**Diseño maqueta de pruebas**																		2										
T4.2	**Diseño plan de pruebas**																		2	2									
T4.3	**Despliegue de pruebas**																					2	2	2					
T4.3.1	Funcionamiento básico																						1						
T4.3.2	Ante caída de enlace																							1					
T4.4	**Análisis y valoración de resultados**																							2					
H5	**Finalizacion pruebas**																								X				
FASE 5	**Gestión y documentación**																									3	3	3	
T5.1	Puesta en marcha y seguimiento																									2	2		
T5.2	Redacción documentación																									2	2		
H6	**Entrega memoria**																										X		

Ilustración 32: Cronograma Gantt

9 Resumen de costes

A continuación, se presentan los costes producidos por la realización de la actividad correspondiente a la producción de este proyecto donde dichos costes se separan en directos e indirectos.

9.1 Costes directos

Costes atribuidos al proyecto:

Horas internas

Teniendo en cuenta las 28 semanas de trabajo, de una duración del proyecto de 213 días se ha estimado 151 días de trabajo [19]. Teniendo en cuenta el promediado de horas invertidas se han considerado 2 horas diarias, por lo que resultaría en 302 horas. Para el proyecto se han adjudicado un Ingeniero Junior y un Ingeniero Senior con los costes unitarios de 15 y 20 respectivamente.

Tabla 11: Costes horas internas

Cargo	Coste unitario	Horas	Coste (€)
Ingeniero Junior	15	302	4.530,00
Ingeniero senior	20	151	3.020,00
TOTAL (IVA Incluido)			7.550,00

Subcontrataciones

Respecto a servicios contratados a otras empresas, en este caso no ha sido necesario hacerlo, pero para proyectos de este tipo más amplios podría ser un coste para considerar.

Tabla 12: Costes subcontrataciones

Servicio	Coste unitario	Horas	Coste (€)
N/A	0	0	0,00
TOTAL (IVA Incluido)			0,00

9.2 Costes indirectos

Amortizaciones

Las amortizaciones tanto para equipamiento software como para material físico corresponden a la representación de cuanto ha sido el coste por su uso a lo largo del proyecto. Para el cálculo del coste unitario para cada una de las amortizaciones se ha tenido en cuenta el coste total del producto y el tiempo de vida estimado por el fabricante.

Amortizaciones de software

El software utilizado a lo largo del proyecto a pesar de ser bastante variado, todos ellos son de uso gratuito, por lo que el gasto total en amortización de software será de 0.

Tabla 13: Costes amortizaciones de software

Software	Coste (€)
Visual Studio Code	0,00
NetScanTools (Beta)	0,00
Microsoft Word	0,00
Microsoft Excel	0,00
TOTAL (IVA Incluido)	0,00

Otras Amortizaciones

A continuación, se incluye el equipamiento utilizado a lo largo del proyecto. Entre ellos se encuentra un ordenador personal portátil, y dos equipos informáticos cada uno con su coste unitario correspondiente a su coste total y vida útil. Se ha estimado un tiempo de uso de 302 horas ya que este equipamiento ha estado en uso durante todo el proyecto

Tabla 14: Costes otras amortizaciones

Componente	Coste unitario	Horas	Coste (€)
Ordenador Portátil	0,08	302	24,16
Equipo Informático 1	0,08	302	24,16
Equipo Informático 2	0,07	302	21,14
TOTAL (IVA incluido)			69,46

9.3 Coste total

Considerando todos los gastos del proyecto, se ha calculado un subtotal de 7.619,46 € (IVA incluido)

Tabla 15: Costes totales

Partida	Coste (€)
Horas internas (IVA Incluido)	7.550,00
Amortizaciones software	0,00
Otras amortizaciones	69,46
Subcontrataciones (IVA Incluido)	0,00
SUBTOTAL	**7.619,46**
Costes indirectos (IVA Incluido)	**380,973**
TOTAL	**8.000,433**

Además de los cálculos realizados anteriormente, también se ha incluido un 5% de costes indirectos los cuales no son atribuibles únicamente a un proyecto específico como podría ser el coste por consumo eléctrico entre otros. Con todo ello los costes totales son de 8.000,433 (IVA Incluido).

10 Conclusiones

En este Trabajo de Fin de grado, se ha desarrollado una solución con el fin de realizar una atención ante la caída de enlaces en una red de comunicaciones haciendo uso de la tecnología DPDK (Data Plane Development Kit). Se ha llevado a cabo mediante el concepto de la virtualización de hardware, mediante la cual se ha posibilitado el realizar un procesado eficiente del tráfico junto con dicha reacción ante fallos de enlace.

Durante la realización de las fases de este proyecto se han indo completando los objetivos planteados al inicio de este. Tanto el diseño como la implementación de la solución han seguido los pasos definidos y cumpliendo con las metas y características descritas.

A lo largo del proyecto se han realizado diversas pruebas de rendimiento, en ellas se ha podido confirmar que haciendo uso de la tecnología DPDK además de optimizar ligeramente el procesado de paquetes, se ha producido una mejora de hasta un 87,5% en los tiempos de recuperación de un enlace respecto a la configuración básica de un direccionamiento en Linux. De esta forma asegurando una mayor estabilidad de la red diseñada y la calidad de servicio de extremo a extremo para los usuarios de una red de telecomunicaciones.

Además de haber obtenido los resultados previstos, esta solución contribuye a la mejora de las redes definidas por software. Principalmente hacia un modelo más robusto y eficiente haciendo uso del concepto del PDP (Planos de Datos Programables). Este concepto permite una mayor flexibilidad en la gestión de las redes de forma centralizada, así como la posibilidad de disponer de una configuración más personalizada en los dispositivos de la red.

Resumiendo, este proyecto además de proporcionar una solución efectiva para la gestión de la caída producida en un enlace también posibilita el desarrollo e implementación de nuevas tecnologías dentro de las redes SDN junto con el plano de datos programable. De esta forma planteando DPDK como una alternativa ante una variedad de tecnologías tanto presentes como futuras que serán utilizadas en estos escenarios.

Bibliografía

[1] *I2T Research Group*. Available: https://i2t.ehu.eus/es.

[2] *Software-Defined Networking (SDN) Definition - Open Networking Foundation*. Available: https://opennetworking.org/sdn-definition/.

[3] F. Hauser *et al*, "A survey on data plane programming with P4: Fundamentals, advances, and applied research," *Journal of Network and Computer Applications*, vol. 212, pp. 103561, 2023.

[4] N. Katta *et al*, "HULA: Scalable Load Balancing Using Programmable Data Planes," .

[5] *P4 – Language Consortium*. Available: https://p4.org/.

[6] *P4 – APS Networks*. Available: https://www.aps-networks.com/technology/p4/.

[7] Anonymous "P4 16 Language Specification version 1.2.4," .

[8] *Barefoot Networks' New Chips Will Transform the Tech Industry | WIRED*. Available: https://www.wired.com/2016/06/barefoot-networks-new-chips-will-transform-tech-industry/.

[9] *P4Runtime Specification*. Available: https://p4.org/p4-spec/p4runtime/main/P4Runtime-Spec.html.

[10] *Stratum - Open Networking Foundation*. Available: https://opennetworking.org/stratum/.

[11] *2. Overview — Data Plane Development Kit 24.03.0-rc2 documentation*. Available: https://doc.dpdk.org/guides/prog_guide/overview.html.

[12] *Priv rings*. Available: https://en.wikipedia.org/wiki/File:Priv_rings.svg.

[13] *32. Intel Virtual Function Driver — Data Plane Development Kit 24.03.0-rc1 documentation*. Available: https://doc.dpdk.org/guides/nics/intel_vf.html.

[14] *7. Poll Mode Driver — Data Plane Development Kit 16.04.0 documentation*. Available: https://doc.dpdk.org/guides-16.04/prog_guide/poll_mode_drv.html.

[15] *DPDK - Installation and Optimization - HackMD*. Available: https://hackmd.io/@sohailanjum97/SkWE46ywu.

[16] *Wireshark · Go Deep*. Available: https://www.wireshark.org/.

[17] *MobaXterm free Xserver and tabbed SSH client for Windows*. Available: https://mobaxterm.mobatek.net/.

[18] *Ubuntu 23.10.1 (Mantic Minotaur)*. Available: https://releases.ubuntu.com/mantic/.

[19] *Business Days Calculator – Count Workdays*. Available: https://www.timeanddate.com/date/workdays.html?d1=1&m1=12&y1=2023&d2=30&m2=6&y2=2024&ti=on&.

ANEXO I

MANUAL DE INSTALACION: DPDK

INDICE

1. Definición características del equipo

1.1. Motherboard:

Dell Precision T700

1.2. CPU:

Intel(R) Core(TM) i7-4770 CPU @ 3.40GHz

1.3. RAM

Hynix HMT41GU6MFR8C-PB 8GB 1600MHz RAM x2

1.4. Tarjeta de red:

Ethernet controller: Intel Corporation I350 Gigabit Network Connection (4 port)

2. Manual de instalación

1.5. Descargar DPDK:
https://fast.dpdk.org/rel/dpdk-23.11.tar.xz

1.6. Extraer (se recomienda hacerlo en /home):
$ tar xf dpdk-23.11.tar.xz

1.7. Instalación de paquetes:
$ sudo apt install build-essential meson python3-pyelftools libnuma-dev pkgconf

Nota: No será necesario instalar ninja / ninja-build ya que este se instalará con mesón

1.8. Entramos en el directorio de DPDK:
cd dpdk-23.11

1.9. Build en el directorio DPDK:
$meson -Dexamples=all build
$ ninja -C build
$ cd build
$ sudo ninja install
$ sudo ldconfig

1.10. Comprobamos los puertos disponibles:
$ dpdk-devbind.py -s

1.11. Apagamos las interfaces que queramos usar para DPDK
$ ifconfig [puerto] down

1.12. Preconfiguración vfio:
- Comprobar que la virtualización está habilitada en la bios
- Editar la siguiente línea en /etc/default/grub:

  GRUB_CMDLINE_LINUX_DEFAULT = “quiet intel_iommu=on”

  sudo update-grub

  Nota: En caso de que se sospeche que el equipo no permita iommu, en la documentación sobre los drivers en DPDK [2] se explica un método no-iommu con el que DPDK puede funcionar perfectamente.

1.13. Bindeo de los puertos con vfio:
$ sudo modprobe vfio-pci
$ dpdk-devbind.py --bind=vfio-pci [puerto]

(La siguiente línea permitirá el bindeo permanente en el equipo en el boot)

$ sudo driverctl set-override [port] vfio-pci

1.14. Comprobamos el estado de los puertos configurados
$ dpdk-devbind.py -s

1.15. Configuración de hugepages:
$ dpdk-hugepages-py -p 1G --setup 3G

Nota: tener en cuenta que los valores incluidos dependerán de las características de los programas a ejecutar, para un comienzo 2G de 1G será suficiente.

1.16. Ejecución programa de muestra:
$ sudo ./dpdk-helloworld -l 0-1 -n 2

Nota: -l indicara el número de núcleos a utilizar y -n la cantidad de bancos de memoria disponibles (en este caso 2)

*Nota: Sera interesante ejecutar el comando *htop* mediante el cual podremos analizar la reserva correcta de las hugepages y del uso de CPU al ejecutar programas de testeo.

3. Compilación

1.17. Crear directorio de trabajo

1.18. Ejecutar:
$ meson init

1.19. Editar código .c

1.20. Comprobar la configuración en el archivo mesón.build:
EJ base:

```
Project('myExample', 'c', version : '0.1', default_options: ['warning_level=3'])

dpdk = dependency('libdpdk')
exe = executable('main', 'main.c', install: true, dependencies: dpdk)
allow_experimental_apis = true
sources = files('myexample.c')
test('basic',exe)
```

1.21. Ejecutar:
$meson setup build
$ cd build

1.22. Ejecutar:
$ ninja

4. Referencias manual

[1] https://youtu.be/0yDdMWQPCOI

Nota: La instalación de DPDK en si comienza en el minuto 7. Todos los pasos anteriores únicamente servirán en caso de querer realizar la instalación en una máquina virtual.

[2] https://doc.dpdk.org/guides-21.11/linux_gsg/linux_drivers.html